Parallelisierte Simulationsprozesse für virtuelles Prototyping in der Automobilindustrie

Von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Carolo – Wilhelmina zu Braunschweig

zur Erlangung der Würde

eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte Dissertation

von: Mathias Hommel

Geburtsname: Müller

aus (Geburtsort): Berlin

eingereicht am: 16. Januar 2006

mündliche Prüfung am: 7. Juni 2006

Referenten: Professor Dr.-Ing. W. Schumacher, TU Braunschweig
Professor Dr.-Ing. P. Adamis, TU Clausthal

2006

Gewidmet allen Mädchen, Frauen und Müttern
für die Liebe, Kraft und Hoffnung, die sie uns geben.

Vorwort

Die vorliegende Arbeit basiert auf der Auswertung von Ergebnissen des Projektes „Gesamtfahrzeugsimulation hybridischer Antriebstränge“ der Konzernforschung der Volkswagen AG in Wolfsburg. Dieses Projekt wurde von mir seitens Volkswagen als Projektleiter allein bearbeitet, definiert und koordiniert.

Herrn Professor Dr.-Ing. Walter Schumacher, Leiter des Institutes für Regelungstechnik an der Technischen Universität Braunschweig und Referator, gilt mein besonderer Dank für die intensiven Diskussionen, die stets aufschlußreich und weiterführend waren.
Herrn Professor Dr.-Ing. Panagiotis Adamis, Honorarprofessor am Institut für Tribologie und Energiewandlungsmaschinen der Technischen Universität Clausthal, mein ehemaliger Abteilungsleiter und Koreferator, gilt mein besonderer Dank, für die konstruktiven Diskussionen und die Ermutigung, diese Promotion zu schreiben.
Bei Herrn Professor em. Dr.-Ing. Dr. h. c. Werner Leonhard, Emeritus des Institutes für Regelungstechnik an der Technischen Universität Braunschweig, bedanke ich mich auf das herzlichste für den Vorsitz.

Im Rahmen des mir zur Verfügung stehenden Budgets suchte ich Unterstützung bei Kollegen und bediente mich der Dienstleistungen verschiedener Fremdfirmen. Aus diesem Grund möchte ich recht herzlich allen beteiligten Mitarbeitern der Volkswagen AG, der Audi Electronic Venture GmbH in Ingolstadt, der Lineas Automotive GmbH in Wolfsburg und der Extessy AG in Wolfsburg danken für die sehr gute Zusammenarbeit. Mein besonderer Dank gilt Herrn Borgs und Herrn Bulakhov der Lineas Automotive GmbH für die vielen, tiefgründigen, fachlichen Diskussionen und für die Unterstützung bei der Modellbildung sowie bei der Inbetriebnahme des Clusterrechners.

Ein besonderer Dank ist gerichtet an meine Eltern für die sorglose Kindheit und die stetige Förderung meiner Entwicklung sowie an meinen Bruder Michael, der mich für die Mikrorechentechnik begeisterte.

Weiterhin möchte ich meiner Frau Susanne und meinen Kindern Lena und Frieder danken für die Geduld und für die Unterstützung, die sie über die Jahre und insbesondere in der Zeit des Verfassens dieser Arbeit mit mir hatten.

Darüber hinaus möchte ich allen meinen Lehrern danken, die mich auf meinem Lebensweg begleitet haben, insbesondere bei Prof. Dr.-Ing. Michael Roth † (TU Ilmenau) und Prof. Dr.-Ing. em. Joachim Holtz (Univ. Wuppertal).

Mathias Hommel
Wolfsburg, 10. Januar 2006

Die Meinungen, Ergebnisse und Schlüsse in dieser Dissertation sind nicht notwendigerweise die der Volkswagen AG.

Inhaltsverzeichnis

Diese Dissertationsschrift ist nach denen in [Duden 1980] beschriebenen, im Jahr 2003 gültigen Regeln der deutschen Rechtschreibung verfaßt.

Bildnachweise: Internet: S.54 (u.), 55

0 Glossar und Abkürzungsverzeichnis

Abtastrate…
…in Zusammenhang mit Controller: Zeitangabe, in welchem zeitlichen Abstand der Algorithmus eines digitalen Controllers von neuem einschließlich Dateneinlesen und Ausgabe abgearbeitet wird.
…in Zusammenhang mit der Simulatorkopplung, EXITE oder Kommunikation: Die Abtastrate gibt an, in welchem zeitlichen Abstand (in Sekunden) ein Datenaustausch zwischen den Kommunikationspartnern erfolgt, d.h. ein Aktualisieren der Interfacedaten.

In beiden Zusammenhängen muß eher von Abtastzeit gesprochen werden als von einer Rate (Angabe pro Zeiteinheit).

ABS
Abkürzung für Antiblockiersystem, ein mechatronisches System innerhalb der Bremsanlage eines Kraftfahrzeuges, welches bei sehr starker Betätigung der Bremse die Räder im Schlupf hält ohne zu blockieren. Auf rauhen und mäßig rauhen Straßenbelägen wird so eine Verkürzung des Bremsweges erreicht bei gleichzeitigem Aufrechterhalten der Lenkbarkeit des Fahrzeuges.

API
engl.: Application Programming Interface: Interface für die Anbindung benutzerdefinierter Software an bereits bestehende, meist kommerzielle Software

ASR
Abkürzung für Antriebsschlupfregelung, ein mechatronisches System, welches in die Bremsanlage oder in die Motorsteuerung eines Kraftfahrzeuges eingreift, um ein Durchdrehen der Räder bei zu starker Fahrpedalanforderung (i. allg. mit „Gas geben“ bezeichnet) zu verhindern.

AUX
engl.: Auxiliaries; Regelstrecke der Nebenverbraucher im Kfz (z. Bsp. Heizung, Klimaanlage, Licht)

BAT
Modell der Regelstrecke Batteriesystem, bestehend aus der Traktionsbatterie, den Schützen und den Sensoren

BMS
Batteriemanagementsystem oder Batteriemonitoringsystem: Steuergerät und/oder darauf ablaufender Steuergerätealgorithmus für die Überwachung des Batteriezustandes und Ansteuerung der Aktuatorik eines Batteriesystems (z. Bsp. Schütze, Lüfter, Temperatursensoren)

CAE
engl.: Computer Aided Engineering; Computer unterstütze Ingenieursarbeit, Entwurfsarbeit

CHS
Modell der Regelstrecke Chassis, bestehend aus den Komponenten Räder, Achsen und Bremssystem.

Controller
Elektronisches Steuergerät im Fahrzeug zur Ansteuerung einer oder mehrerer Komponenten basierend auf einem oder mehreren Mikrokontroller; Controller wird auch als Synonym für den Funktionsalgorithmus eines elektronischen Steuergerätes verwendet.

CORBA
engl.: Common Object Request Broker Architecture; CORBA ist eine objektorientierte, plattformunabhängige Kommunikationsarchitektur innerhalb des ISO-OSI-Schichtenmodells, die die Koordination, Synchronisation und Kommunikation zwischen Objekten realisiert..

CRC
engl.: Cyclic Redundancy Code; Form einer Checksumme, die durch mathematische Operationen über binäre Daten gebildet wird, um Fehler bei Datenübertragungen zu erkennen.

DAE
engl.: Differential Algebraic Equation; System differential algebraischer Gleichungen, bei denen zusätzlich zu den Differentialgleichungen (DGL) noch rein algebraische Nebenbedingungen in Form von algebraischen Gleichungen (AGL) eingebracht werden. Die allgemeinste Form einer DAE ist eine implizite DGL der Form

$$f(\dot{x}, x, t) = 0 \qquad \text{(Gl. 0.1)}$$

Diese Gleichung ist nur dann lokal nach $\dot{x}$ auflösbar, wenn die partielle Ableitung $f_{\dot{x}}$ regulär ist. In diesem Fall kann die implizite Form aus (Gl. 0.1) umgeschrieben werden in eine semi-explizite DAE:

$$\begin{aligned} \text{DGL:} \quad & \dot{\underline{x}}_1 = \underline{f}_1(\underline{x}_1, \underline{x}_2, t) \\ \text{AGL:} \quad & 0 = \underline{f}_2(\underline{x}_1, \underline{x}_2, t) \end{aligned} \qquad \text{(Gl. 0.2)}$$

mit $\underline{x}_1$ :: Vektor der differentiellen Variablen
$\underline{x}_2$ :: Vektor der algebraischen Variablen

Das Bestimmen von $\underline{x}_1(t)$ und $\underline{x}_2(t)$ bei vorgegebenen, zu den Gleichungen (Gl. 02) konsistent gewählten Anfangswerten $\underline{x}_1(t_0)$ und $\underline{x}_2(t_0)$ bezeichnet man als das Lösen von DAEs, welches eine nichttriviale Aufgabe ist.

DCW
Modell des bidirektionalen DC/DC-Wandlers zwischen den Systemen Traktionsbatterie und Bordnetzbatterie

DLL
engl.: Dynamic Link Library; Programmteil, der zur Laufzeit (dynamisch) vom aufrufenden Programm aktiviert wird. Diese Methode spart Arbeitsspeicher (RAM). Eine DLL kann unabhängig vom aufrufenden Programm entwickelt und gewartet werden. DLL-Files haben oft die Namenserweiterung dll.

DRV
engl.: Driver; Fahrermodell, in welchem die durch den Fahrzyklus vorgegebene Fahrzeugsollgeschwindigkeit mit der simulierten Fahrzeugistgeschwindigkeit verglichen wird und aufgrund dieser Regeldifferenz die Stellgrößen Fahrpedalwert und Bremspedalwert generiert werden.

DSG
Direktschaltgetriebe der Volkswagen AG; ein zugkraftunterbrechungsfrei schaltendes, automatisches Schaltgetriebe

Echtzeit
Mit Echtzeit ist in dieser Arbeit nicht die Eigenschaft echtzeitfähiger Systeme gemeint, die die Bedingungen an eine sogenannte harte Echtzeit erfüllen und innerhalb einer definierten Zeit ein garantiertes Antwortverhalten besitzen, sondern die Eigenschaft einer Simulation, die Berechnung des Verhaltens eines Modells in genau der gleichen Zeit durchzuführen, wie sich das Modell in der Realität verhalten würde. Wenn eine Simulation n-mal „schneller" als Echtzeit ist ($n > 1$), dann ist die gemessene Simulationsdauer bzw. vergangene Rechenzeit eines bestimmten Simulationszeitintervalls kürzer als dieses Simulationszeitintervall.

ECU
engl.: Electronic Control Unit; Elektronisches Steuergerät im Fahrzeug zur Ansteuerung einer oder mehrerer Komponenten basierend auf einem oder mehreren Mikrokontroller

EMA
Modell des elektromotorischen Antriebs, bestehend aus dem Modell des Elektromotors und der Hardwareschaltungen für die Ansteuerung (z. Bsp. Wechselrichter)

EMA-ECU
Steuergerät des elektromotorischen Antriebs bei Elektro- und Hybridfahrzeugen

EXITE
Werkzeug und Programmierpaket der Fa. Extessy AG, Wolfsburg, zum Koppeln von Simulatoren.

EXITE-Interface
Das EXITE-Interface definiert die Anzahl der auszutauschenden Daten während der Simulation. Das EXITE-Interface ist durch eine Interfacebeschreibung, der Abtastrate sowie dem vollständigen Namen, einschließlich aller Packagenamen, eindeutig identifiziert und dient als Grundlage des Datenaustauschs zwischen den Client- und Server-Modellen. Es wird zwischen Input- und Output-Ports unterschieden. Die Daten des EXITE-Interfaces werden im EXITE-Server zentral für alle im Rechnerverbund beteiligten Modelle abgelegt. (In den Versionen von EXITE bis zur Version 1.3.4 wird der Begriff Slave mit Client gleichgesetzt und Master mit Server.)

EXITE Cluster-Service
Der EXITE Cluster-Service ist ein installierter EXITE-Service, der als NT-Netzwerkdienst im Hintergrund auf einem zentralen Server gestartet wird. Er verwaltet die Simulationssession und verbindet die Kommunikationspartner untereinander. Den EXITE Cluster-Service gibt es genau einmal innerhalb eines Simulationsclusters. Synonyme für EXITE Cluster-Service sind auch Cluster-Service, Cluster-Server oder EXITE-Server.

EXITE Model-Service
Der EXITE Model-Service wird ebenso wie der EXITE Cluster-Service im Hintergrund als NT-Dienst gestartet. Er muß auf jeden Rechner gestartet werden, wo ein Teilmodell abgearbeitet werden soll und ist verantwortlich für das Suchen nach und das automatische Starten von Modellen aus dem (lokalen) Model-Repository. Synonyme für EXITE Model-Service sind auch Model-Service und Model-Server.

EXITE-Master-Modell
Master-Modell bzgl. der Modellkopplung durch EXITE bedeutet, daß die verteilt gerechneten Teilmodelle im betrachteten Modell als Stellvertretermodelle eingebunden sind. Somit übernimmt der Master die Rolle des Clients in der Kommunikation und wird deshalb auch als **EXITE-Client** oder **Clientmodell** bezeichnet.

EXITE-Slave-Modell
Das Slave-Modell im Sinne EXITE ist das ausgelagerte Teilmodell und übernimmt die Rolle des Servers in der Kommunikation und wird als **EXITE-Server** bezeichnet. Das verlagerte Teilmodell kann auf dem gleichen Rechner oder auf einen anderen Rechner, in der gleichen oder einer anderen Umgebung (Modellierungstool) abgearbeitet werden.

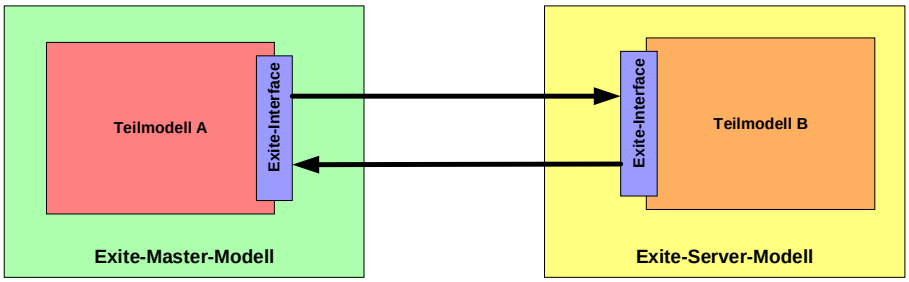


Abbildung 0.1: Darstellung zu den Begriffen Master- und Slave-Modell

GTR
Modell der Regelstrecke Getriebe einschließlich Sensorik und Aktuatorik

HIL
engl.: Hardware-in-the-Loop; Methode zur Funktionsprüfung von Steuergerätehard- und -software, wobei das zu untersuchende Steuergerät mit einen Simulator verbunden ist, welcher die anzusteuernde Regelstrecke in harter Echtzeit simuliert.

HMI
engl.: Human Machine Interface; Modell der Schnittstelle des Fahrers (Mensch) mit dem Fahrzeug (Maschine). Bezüglich der vorliegenden Gesamtfahrzeugsimulation vereint das HMI-Modell alle die Schalter und Bedienelemente, die sonst keinem Antriebstrangaggregat zugeordnet werden können, wie z. Bsp. das Zündschloß mit dem Zündanlaßschalter.

ISO-OSI-Schichtenmodell
engl.: International Standard Organization - Open Systems Interconnection; Standardisiertes Modell zum Aufbau von Verbindungen zu entfernten Informationsobjekten über physikalische Übertragungskanäle.

LAN
engl.: Local Area Network; Datenübertragungsnetzwerk, welches für lokale Zwecke, z. Bsp. innerhalb eines Gebäudes oder eines Standortes eines Unternehmens konzipiert ist.

LANai-Chip
Der LANai-Chip ist ein von der Firma Myricom, USA entwickelter, programmierbarer Kommunikationschip, der das Interface des Myricom-Netzwerkarten-eigenen system area network (SAN) bedient. Er enthält unter anderem einen Mikroprozessor, LANai-core genannt. Eine vollständige Beschreibung kann dem Internet entnommen werden.
(http://www.myri.com/vlsi/LANai7.pdf)

LTI
engl.: Linear and Time Invariant; Bezeichnung in der Systemtheorie für die Eigenschaft von Übertragungsfunktionen, deren Differentialgleichungen linear und zeitlich invariant sind.

NEDC / NEFZ
engl.: New European Driving Cycle bzw. deutsch für Neuer Europäischer Fahrzyklus; Ein synthetischer Fahrzyklus von der europäischen Automobilindustrie definiert, für die Bewertung und den Vergleich von Kraftfahrzeugen. Er setzt sich zusammen aus einem Stadtanteil (ECE), der drei Mal wiederholt wird, und einem Überlandanteil (EUDC):

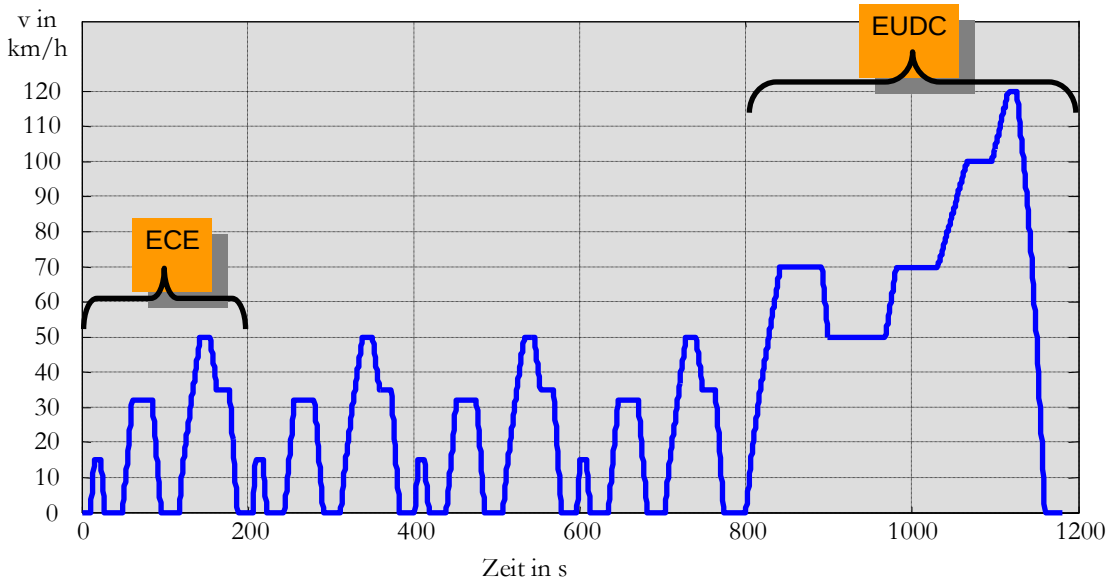


Abbildung 0.2: Definition des zeitlichen Geschwindigkeitsverlaufs v(t) des NEDC

MPI
engl.: Message Passing Interface; MPI ist eine standardisierte Kommunikationssteuerschicht im ISO-OSI-Schichtenmodell der Datenkommunikation. MPI wurde speziell für die nachrichtenbasierten Kommunikation der parallelen Datenverarbeitung in heterogenen Rechnernetzwerken entwickelt [MPI 1998, MPI 2003].

Myrinet
Myrinet ist der eingetragene Markenname der Netzwerkkarte der Firma Myricom, USA. Die Fa. Myricom (www.myri.com) wurde aus Mitgliedern der amerikanischen Organisationen ISI ATOMIC group und CalTech Mosaic group gegründet. Aus letzterer wurden LAN-basierte Komponenten des CalTech Mosaic supercomputers für einen Vorläufer der Myricom-Netzwerkkarte entnommen. Es existieren Netzwerkkarten sowohl auf Basis kupferner wie auch auf optischer Übertragung. Myrinet unterstützt die Protokolle TCP, GM, MPI.

ODE
engl.: Ordinary Differential Equation; gewöhnliche Differentialgleichung, bei der nur die Ableitung nach einer reellen Variablen auftritt. Die Ordnung der ODE ist durch die höchste vorkommende Ableitung gegeben. ODEs beliebiger Ordnung lassen sich immer in ein System von ODEs erster Ordnung umwandeln. Lineare ODEs sind linear in der unbekannten Funktion und ihren Ableitungen.

ODE n-ter Ordnung: $$y^{(n)}(t) = f(t, y(t), y'(t), y''(t), ..., y^{(n-1)}(t))$$ (Gl. 0.3)

Package
EXITE-Package: Windowsverzeichnisse oder Ordner
Modelica-Package: Teilbibliothek innerhalb der objektorientierten Modelicabibliothek
RAM
engl.: Random Access Memory; Schreib-Lesespeicher mit wahlfreiem Zugriff

s-function:
Als s-function wird ein allgemeines Interface für die Ankopplung an die blockorientierte Modellierung in Simulink bezeichnet. Es wird unterschieden zwischen vorkompilierten s-functions, nicht-vorkompilierten und s-functions in der Matlab-Scriptsprache [Matlab 2002].

PRT
Modell der Regelstrecke Powertrain; Der Powertrain (deutsch: Antriebstrang) ist die momentenerzeugende und -verteilende Einheit im Kfz, die bei einem Hybridantrieb z. Bsp. aus dem Verbrennungsmotor und dem Getriebe sowie dem elektrischen Traktionssystem und dem elektromotorischen Antrieb besteht.

SIL
engl.: Software-in-the-loop; Simulation von zu entwickelnden Controlleralgorithmen in einer reinen Simulationsumgebung, d.h. auch die Regelstrecke wird modelliert

Simulator
Softwarewerkzeug, mit dem man Simulationen durchführen kann. Typischerweise enthalten diese Werkzeuge auch das Tool der Modellierung.

SOC
engl.: State Of Charge; Ladezustand einer Batterie in Prozent, der gegeben ist durch das Verhältnis von aktueller Ladung und nomineller, maximal möglicher Ladung

SOP
engl.: Start of Production; Zeitpunkt des Produktionsanlaufes

VKM
Verbrennungskraftmaschine; Modell des Verbrennungsmotors eines Kraftfahrzeuges

VMU
Vehicle Management Unit; Fahrzeugsteuergerät, in welchem die Strategien des Hybrid- und Energiemanagements hinterlegt sind.

VSP
engl. Vehicle Simulation based on Physics; Bezeichnung einer Simulations- und Entwicklungsumgebung für die physikalische Gesamtfahrzeugsimulation in der Konzernforschung der Volkswagen AG

WAN
engl.: Wide Area Network; Datenübertragungsnetzwerk, welches aufgrund spezieller Methoden (z. Bsp. Adressdecodierung) für große, raumgreifende Rechnernetze ausgelegt ist.

1 Einleitung – Das virtuelle Prototyping

In Zeiten der Verknappung von Energieressourcen wird nach alternativen Konzepten in der Antriebstechnik von Kraftfahrzeugen gesucht, die bei vergleichbarem Komfort und vergleichbarer Fahrleistung einen verbrauchsmindernden Einfluß haben. Hybridfahrzeuge zeigen dabei ein signifikantes Potential auf, innerhalb der vorgegebenen ökonomischen, politischen und infrastrukturellen Randbedingungen den Kraftstoffverbrauch der Flotte zu senken.
Derzeit führen alle großen Automobilhersteller Machbarkeitsstudien durch, die auf den einschlägigen Messen und Konferenzen vorgestellt werden [Hommel 2005]. Seit einigen Jahren werden von verschiedenen Fahrzeugherstellern Serienlösungen angeboten. So haben zum Beispiel Toyota und Honda bereits 2002 mit der Serienproduktion begonnen. Für die nächsten fünf Jahre wird ein weiter wachsender Markt erwartet.

Bei dem Betreiben dieser hybridischen Antriebstränge wird eine übergeordnete Steuerung benötigt, die die kraftstoffsparende Betriebsstrategie umsetzt und die Leistungsaufteilung zwischen Verbrennungsmotor und Elektromotoren vorgibt. Diese Steuerung wird häufig als sogenannter Hybridmanager oder Energiemanagement bezeichnet. Bei der Entwicklung der Steuerungssoftware eines Controllers (s. Glossar) und so auch des Hybridmanagers wird im Allgemeinen nach dem bekannten V-Modell (s. Abbildung 1.2) vorgegangen und getestet. Das Verwenden eines Vorgehensmodells kann das Testen erheblich vereinfachen und die Qualität der Software maßgeblich positiv beeinflussen [V-Modell 1997].
Das V-Modell löst das sogenannte Wasserfallmodell der Softwareerstellung (s. Abbildung 1.1), welches ein reines Phasenmodell ist, ab, da dieses den Nachteil hat, daß das Testen der Software selbst in einer zu späten Phase erfolgt.

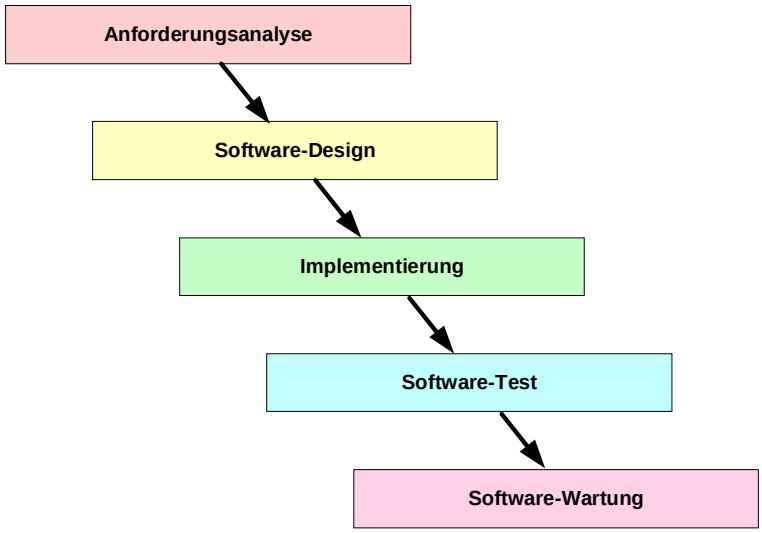


Abbildung 1.1: Das Phasenmodell oder sogenannte Wasserfallmodell der Softwareentwicklung. Der Softwaretest erfolgt in einer sehr späten Phase. In neueren Publikationen wird dieses Modell so abgeändert, daß Iterationen in Zyklen mit nächst höheren Schichten möglich sind.

Das V-Modell ist ein Entwicklungsstandard für IT-Systeme der Bundesbehörden in Deutschland und wurde im Auftrag des Bundesministeriums für Verteidigung (BMVg) entwickelt. 1992 wurde es vom Bundesministerium des Innern (BMI) für den zivilen Verwaltungsbereich der Bundesbehörden übernommen [V-Modell 2001].

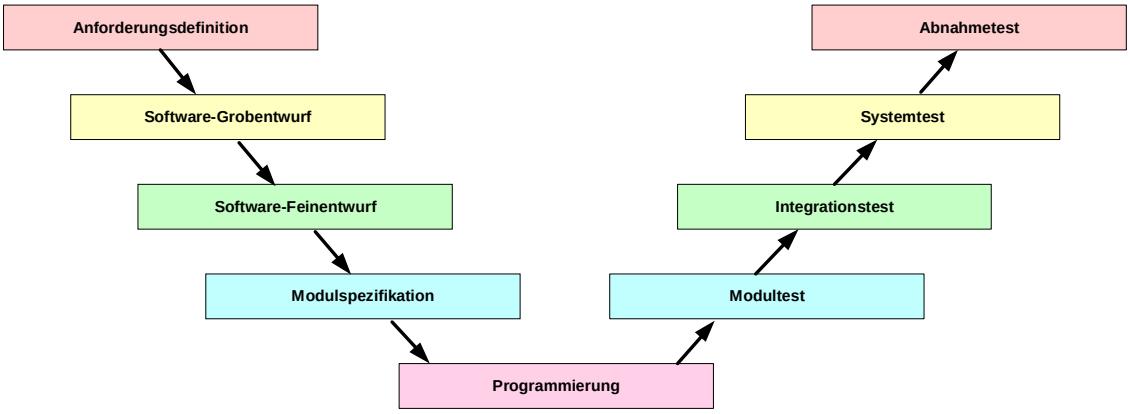


Abbildung 1.2: Das allgemeine V-Modell der Softwareerstellung. Die jeweils farblich gleichen Phasen in beiden Zweigen entsprechen einander bzw. der jeweilige Test prüft die Spezifikation auf dem linken Ast des V-Models ab.

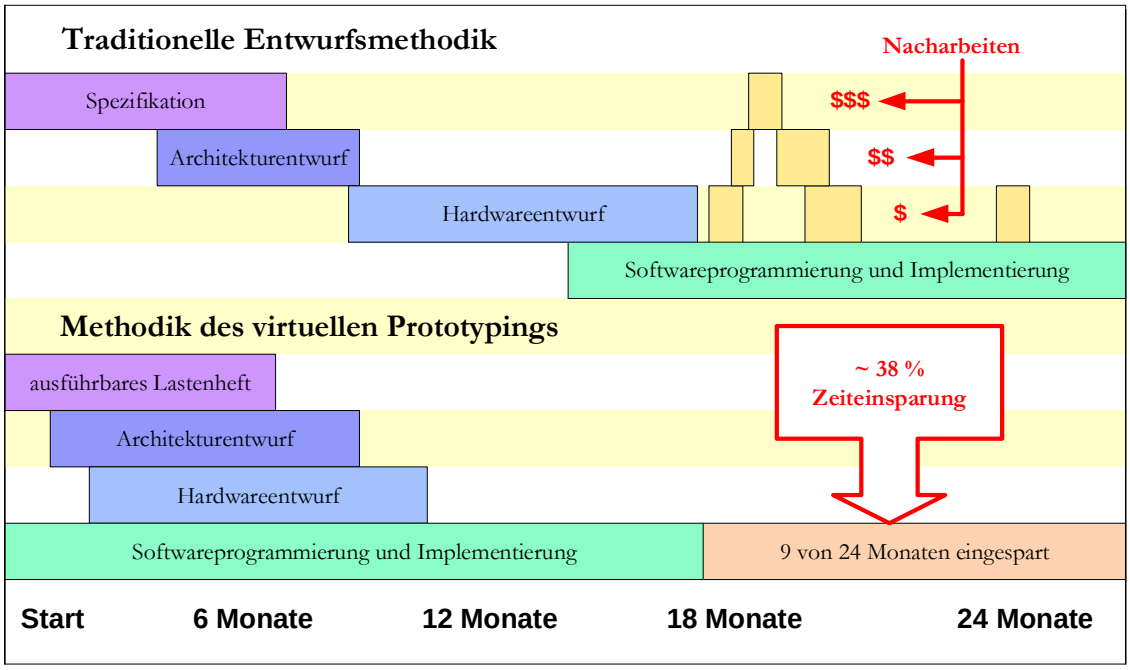
Die systematische Vorgehensweise der Softwareentwicklung nach dem V-Modell hat den Vorteil, daß die Testphasen den Entwicklungsphasen angepaßt bzw. zugeordnet sind. Das V-Modell der Softwareentwicklung wurde in der Automobilindustrie bei der Erstellung von Steuergerätesoftware übernommen und wird heute in vielfacher Weise im Softwareentwicklungsprozeß angewendet [Miegler et al. 2003].

Typischerweise fand bei der Entwicklung mechatronischer Systeme in der Vergangenheit die Entwicklung der Software ***nach*** der Entwicklung einer entsprechenden Hardware statt. Es wurden zum Beispiel erst die integrierten Schaltungen entwickelt und danach erst die Software, die auf diesen Schaltungen abgearbeitet werden sollte [Hellestrand 2005]. Durch diese Vorgehensweise konnten Fehler in der Hardware erst relativ kurzzeitig vor SOP entdeckt und in zeitlich aufwendigen und teuren Iterationen, nämlich der Änderung des Hard- und Softwaredesigns, beseitigt werden. In [Hellestrand 2005] wird dieser Prozeß am Beispiel eines Herstellers für integrierte Hard- und Softwarelösungen für die mobile Kommunikationstechnik erläutert. Auf ganz ähnliche Weise wurden in der Vergangenheit Hybridfahrzeuge entwickelt: Erst wenn die Aggregate einschließlich der Steuerung verfügbar, d.h. gegenständlich waren, konnte ein Integrationstest der entwickelten Fahrzeugsteuerungsalgorithmen erfolgen. Daß heißt, erst auf dem kompletten Antriebstrangprüfstand bzw. im Prototypenfahrzeug konnte mit der eigentlichen Entwicklung der Hybridalgorithmen hinsichtlich ihres Zusammenspiels mit allen relevanten Controllern begonnen werden.

In Zukunft fließen die Entwicklungsphasen von Hard- und Software bei der Entwicklung mechatronischer Systeme immer mehr ineinander über, so daß der rein sequentielle Prozeß der Entwicklung mechatronischer Systeme abgelöst wird durch einen Prozeß der virtuellen Entwicklungsmethodik. Dieser Prozeß wird nach der englischen Bezeichnung Virtual Prototyping oder Frontloading genannt (s. Abbildung 1.3).

Durch Simulationsmethodik des virtuellen Prototypings ist es möglich, wertvolle Entwicklungszeit einzusparen durch paralleles Entwickeln von Hard- und Softwarekomponenten. Neben den positiven Auswirkungen auf das Kosten- und Zeitbudget von Projekten besteht

durch das virtuelle Prototyping und der damit verbundenen Reproduzierbarkeit von Simulationen die Möglichkeit, eine robuste Software zu entwickeln, wie es durch zeitlich begrenzte Applikation und Tests nicht möglich ist.



(nach [Hellestrand 2005])

Abbildung 1.3: Mikroprozessorhard- und softwaredesignzyklus der Fa. VaST Systems Technology, Sunnyvale, USA

Da im Weiteren die Begriffe virtuelles Prototyping und Frontloading immer wieder verwendet werden, sollen beide Begriffe an dieser Stelle kurz definiert werden:

> *„Unter **virtuellem Prototyping** wird die Gesamtheit der Techniken verstanden, die notwendig sind, um die Produktentwicklung weitgehend computerunterstützt durchführen zu können.“* [Barth et al. 2002]

> *„**Frontloading** bezeichnet den Prozeß, die Simulation und die Analyse bereits in der frühen Konzept- oder Konstruktionsphase eines neuen Produktes so zu integrieren, daß möglichst viele wichtige Produktentscheidungen durch virtuelle Versuche abgesichert werden können.“* [Frontloading 2004]

Beide Methoden beschreiben einen Prozeß des CAE der die Lücke schließt zwischen den verschiedenen Softwaretests des rechten Astes des V-Modells und dem Hardware-in-the-loop-Test (HIL-Test), wo das Objekt der Entwicklung als gegenständlicher Prototyp in einer virtuellen Umgebung getestet wird. Wie an Beispielen in [Barth et al. 2002] und in [Frontloading 2004] beschrieben, gewinnt das virtuelle Prototyping bzw. Frontloading gegenwärtig immer mehr an Bedeutung.

Die Simulationen können in verschiedenen Phasen des V-Modells hilfreich sein und es gibt abgestimmt auf die jeweilige Phase im Softwareentwicklungsprozeß spezifische Simulationen. Zu nennen wären je nach Modellausprägung und Detaillierungsgrad zum Beispiel folgende Simulationen:
(entnommen [Kleinwechter et al. 2005])

- ***„physikalisch-konzeptionelle Simulation:*** *Das Modell konzentriert sich auf die mathematisch/physikalische Abbildung des Modellierungsgegenstandes. Es dient der regelungstechnisch/mathematischen Analyse bzw. Lösungsfindung (z. Bsp. Reglerstrategie und -design, Streckenanalyse). Es kann ebenfalls zur Ermittlung und Definition der wesentlichen Zustände einer abzubildenden Funktionalität sowie der möglichen Zustandsübergänge dienen. Das Modell abstrahiert vollständig von Softwareaspekten und erhebt im Falle eines Fahrzeugfunktionsmodells keinen Anspruch auf Vollständigkeit. Im Allgemeinen sind physikalisch-konzeptionelle Modelle zeit- und wertekontinuierlich. Im Entwicklungsprozeß werden physikalisch-konzeptionelle Modelle üblicherweise in der* ***Systemanalyse und dem Systemdesign*** *(Fahrzeugfunktion) bzw. für den* ***Systemtest*** *(Strecke) eingesetzt.*

- ***Verhaltensorientierte Simulation:*** *Verhaltensmodelle konzentrieren sich auf die vollständige Abbildung des Verhaltens einer Funktionalität im algorithmischen Sinne. Dabei können ebenfalls Fehlererkennung bzw. Reaktionen auf Fehler modelliert werden, wie z. Bsp. das verspätete Empfangen eines Signals oder unplausible Eingangswerte. Bei der Modellierung wird die spätere Umsetzung auf eine digitale Laufzeitumgebung durch zeitdiskrete Simulation berücksichtigt, wodurch Auswirkungen der Abtastzeit auf das funktionale Verhalten untersucht werden können. Im Allgemeinen wird eine ideale Umwelt in Bezug auf Rechnerressourcen (Prozessorzeit, Speicherbedarf) angenommen. Es wird z. Bsp. auf den Einsatz von Integer-Datentypen verzichtet. Im Entwicklungsprozeß wird das Verhaltensmodell vorwiegend in der* ***Systemanalyse bzw. dem Systemdesign*** *(Fahrzeugfunktion Detailentwurf) eingesetzt und dient der Verifikation und Validation der System- bzw. Funktionsanforderungen. Hierzu werden sowohl PC-Host-Simulation als auch Rapid Prototyping eingesetzt. Ebenso können Verhaltensmodelle für die* ***Testphase*** *in Form von Strecken- und Testmodellen verwendet werden.*

- ***Implementierungsorientierte Simulation:*** *Das Implementierungsmodell beschreibt eine Fahrzeugfunktion mit dem Ziel der Ableitung einer serientauglichen, ablauffähigen Software. Die Modellierung konzentriert sich dabei auf eine möglichst effektive Abbildung des Algorithmus mit Bezug auf den Ressourcenverbrauch in einem Steuergerät. Hierzu werden üblicherweise Integerdatentypen verwendet, die in der Regel ausgerichtet auf eine konkrete Ziel-Hardware optimiert eingesetzt werden. Das Implementierungsmodell ist die Grundlage für die automatische Generierung von Seriensteuergerätecode. Im Entwicklungsprozeß werden Implementierungsmodelle üblicherweise in den Phasen* ***Software-Design und Implementierung*** *eingesetzt, wobei letztere durch die automatische Codegenerierung größtenteils abgedeckt wird.“*

Die Entwicklung von Steuergerätealgorithmen in einem simulierbaren Prototypenprogramm wird auch als „Digitales Lastenheft“ bezeichnet und setzt eine lückenlose, konsistente Spezifikation der gewünschten Funktionalität voraus. Gerade im Fall des Hybridmanagers, also eines übergeordneten Steuergerätes mit überlagerten Regelstrategien, welches sämtliche Komponenten im Antriebstrang ansteuert, ist diese konsistente Spezifikation notwendig.

Wegen der Komplexität des Eingriffs des Hybridmanagers in fast alle Systeme eines Kraftfahrzeuges, war es in der Vergangenheit nicht möglich, die Software des Hybridmanagers als ausführbares Lastenheft zu spezifizieren. Immer war der Fahrversuch notwendig, um

Funktionen des Hybridmanagers zu definieren, umzusetzen, zu testen und zu applizieren, was mit viel Aufwand verbunden ist und damit einen erheblichen Einfluß auf die Projektdauer und die Kosten hat.

Aufgrund jedoch der heute zur Verfügung stehenden, leistungsfähigen Rechentechnik, ist es mit einer komplexen Antriebstrangsimulation, die die relevanten Systeme wie Batterie, Elektromotor und deren Verhalten berücksichtigt, möglich, schon die meisten Funktionen in Form eines Digitalen Lastenheftes „vorauszuentwickeln", um überlagerte Regelstrategien unter Berücksichtigung aller relevanten mechatronischen Komponenten zu entwickeln. Da mit Hilfe des virtuellen Prototypings Auslegungen mechatronischer Komponenten bewertet werden können, schließt die hier vorgestellte Simulationsmethodik der Gesamtfahrzeugsimulation die Lücke zwischen Konstruktion/Entwurf und HIL-Simulation. Auch, um über zukünftige oder hypothetische Antriebskonzepte Aussagen machen zu können wie zum Beispiel die mögliche Reichweite oder den Verbrauch, bedarf es einer Simulation, die aufgrund der Komplexität Vorteilhafterweise auf einem Rechencluster als sogenannte Clustersimulation durchgeführt wird, damit die hier vorgestellte Simulation handhabbar bleibt. Darüber hinaus sollen die in dieser Gesamtfahrzeugsimulation entwickelten Funktionen der einzelnen Controller und speziell des Hybridmanagers durch automatische Codegenerierung direkt in ausführbaren Steuergerätecode übersetzt werden können, so daß der Prozeß gemäß Abbildung 1.2 von der Spezifikation bis zur Implementierung ein durchgängiger Entwicklungsprozeß für Steuergerätesoftwareentwicklung ist und gemäß der Abbildung 1.3 die Auslegung der Komponenten parallel zu der Entwicklung der Steuerungssoftware erfolgen kann.

In der hier vorliegenden Arbeit wird eine Simulationsmethodik vorgestellt, dies es ermöglicht, Hard- und Softwarekomponenten entsprechend des gewählten Detaillierungsgrades virtuell auszulegen und zu bewerten. Am Beispiel einer Simulation zur Spezifikation von Komponenten und Controlleralgorithmen eines hybridischen Antriebstranges soll diese Simulationsmethodik erläutert werden. Dafür sind Vorüberlegungen notwendig, welche Simulatoren sinnvollerweise bei der Simulation zum Einsatz kommen sollten und auf welche Art und Weise die hard- und softwareseitige Kopplung der Simulatoren realisiert werden kann, um die Handhabbarkeit zu gewährleisten. D.h. verwendete Simulationswerkzeuge und Techniken der Simulatorkopplung beeinflussen die Rechenbarkeit des Modells wesentlich wobei unter Handhabbarkeit nicht nur die Fähigkeit des Rechnersystems zu verstehen ist, komplexe Modelle schneller als Echtzeit auszuführen sondern auch die Beantwortung der Fragen: Wie unkompliziert und schnell sind Modelländerungen möglich und wie vertraut ist der Anwender mit den verwendeten Simulatoren?

Das Ziel der vorliegenden Arbeit ist es zu klären, unter welchen Randbedingungen virtuelles Prototyping in der Automobilindustrie möglich ist. Anhand eines konkreten Beispiels soll diese Fragestellung untersucht werden.

Diese Arbeit fokussiert auf die Antriebsstränge alternativer Antriebskonzepte in der PKW-Automobilindustrie. Jedoch können die Ergebnisse auf andere Domänen in der PKW-Entwicklung übertragen werden und auch auf Entwicklungsziele in der Nutzfahrzeugentwicklung.

Im Kapitel 2 wird in einer Vorbetrachtung auf die verschiedenen Simulatoren und die Möglichkeiten der Modellbildung eingegangen. Die Modellbildung des in der vorliegenden Gesamtfahrzeugsimulation verwendeten hybridischen Antriebstranges wird in Kapitel 3 erläutert. Der Aufbau des Simulationsrechners sowie die hard- und softwareseitigen Kopplungsmechanismen der Simulatoren sind in Kapitel 4 beschrieben. Kapitel 5 ist der Validie-

rung der Gesamtfahrzeugsimulation gewidmet. Dazu sollen die Ergebnisse der Simulation mit Meßdaten des realen Fahrzeugs verglichen werden. Die Zusammenfassung der Ergebnisse und die Schlußfolgerungen sind in Kapitel 6 dargelegt. Als Ausblick werden in Kapitel 7 verschiedene Möglichkeiten der Performancesteigerung genannt.

2 Einleitung zur Modellierung

Die Wahl des geeigneten Werkzeuges für die Modellbildung und die Simulation hängt von verschiedenen Kriterien ab, wie zum Beispiel von der Verfügbarkeit, von den Kosten, vom verwendeten Betriebssystem usw.

In Kapitel 2.1 wird ein Überblick gegeben über die gängigen Modellierungstools für die Modellierung von Regelstrecken, die stets eine Abstraktion der realen Physik sind. Im Fall der vorliegenden Gesamtfahrzeugsimulation erstreckt sich die Regelstrecke über mehrere Domänen (Mechanik, Hydraulik, Elektrotechnik). Speziell für die Antriebstrangmodellierung haben sich einige Tools etabliert, die in Kapitel 2.2 aufgelistet werden. Das Modellieren von Steuergerätealgorithmen unterscheidet sich in einigen Punkten von der Modellierung physikalischer Systeme, wie zum Beispiel die Wahl der systemtheoretischen Beschreibung (kontinuierlich oder diskret). Darauf wird in Kapitel 2.3 eingegangen. Der Unterschied zwischen kausaler und akausaler Modellierung wird in Kapitel 2.4 erläutert. In Kapitel 2.5 wird ein kurzer Überblick über die Modellierung in Modelica/Dymola gegeben. Schließlich werden in Kapitel 2.6 die Kriterien für die Wahl der Modellierungsmethodik vorgestellt und der Aufbau der vorliegenden Gesamtfahrzeugsimulation erläutert.

2.1 Modellierungstools für die Regelstreckensimulation

Für die Simulation unterschiedlichster Anwendungen sind eine ganze Reihe verschiedener Softwarewerkzeuge verfügbar. Im Folgenden werden Werkzeuge vorgestellt, mit denen mechatronische Systeme allgemeiner Art in verschiedenen Domänen (hydraulische, elektrische, mechanische Systeme) simuliert werden können. Die Preise für die Lizenzen bzw. die angegebenen Internetverweise in diesem und im folgenden Kapitel wurden Ende 2004 ermittelt.

ASCL Sim der Firma The AEgis Technologies Group Inc., USA :

- http://www.acslsim.com/products/SIM/sim.htm
- 25 Jahre alte Modellierungs- und Simulationssprache
- Modellierung jeglicher kontinuierlicher Systeme (Multidomänmodellierung)
- basierend auf dem CSSL-Standard (continous system simulation language), einer textuellen Beschreibungssprache, mit der die Differentialgleichungen und Zusammenhänge programmiert werden
- nur auf der grafischen Oberfläche block- und scheinbar objektorientiert
- sehr leichte Portierbarkeit auf unterschiedlichster Entwicklungshardware
- Lizenz: ca. U$ 8.600,- für ein Jahr , Wartung: U$ 1700,-/Jahr oder perpetual license U$15.000,- / 25 Jahre

Dymola der Firma Dynasim, Schweden:

- http://www.dynasim.com/
- objektorientiertes, gleichungsbasiertes Modellieren
- leistungsstark für große Multidomänsysteme
- basierend auf dem offenen Modellierungsstandard Modelica
- Lizenz: ca. € 8.000,-

Extend der Firma Imagine That Inc., USA:

- http://www.imaginethatinc.com/prods_prodline.html
- blockdiagrammbasiertes Modellieren
- einfach aber leistungsstark
- nur für ereignisdiskrete Systeme geeignet
- Lizenz: ca. € 3.300,-

gProms der Firma Process Systems Enterprise Limited, GB:

- http://www.psenterprise.com/products_gproms.html
- Multidomänsimulations- und optimierungstool
- leistungsstarke Handhabung von partiellen Differentialgleichungen
- gleichungsbasiert, textuell
- grafische Darstellung in Blockschaltbildern möglich
- typische Anwendungsgebiete: Chemie, Reaktortechnik
- Lizenz: ca. € 19.000,-

IDA Simulation Environment (IDA SE) der Firma EQUA, Schweden:

- http://www.equa.se/eng.se.html
- gleichungsbasierte Simulationsentwicklung, basierend auf Modelica
- entwickelt speziell für die Domänen Wasser und Abfluß
- begrenztes Berechnungsvermögen
- Lizenz: ca. SEK 25.000,-

IDEAS Gold der Firma IDEAS Simulation Inc., USA:

- http://www.ideas-simulation.com/home.php
- Erweiterung von Extend für die dynamische Simulation kontinuierlicher Systeme
- Insbesondere für hydraulische und thermodynamische Simulation geeignet
- Lizenz: ca. € 16.000,-

MathModelica der Firma MathCore Engineering AB, Schweden:

- http://www.mathcore.com/
- Modellierungs- und Simulationstool für dynamische Multidomänsysteme
- basiert auf dem Anknüpfen von Modelica an Mathematica, welches ein Textverarbeitungs- und Mathematikprogramm ist
- Lizenz: ca. € 10.200,-

Matlab/Simulink der Firma The MathWorks Inc., USA:

- http://www.mathworks.de
- leistungsstarkes Modellierungspaket mit umfangreichen Visualisierungsmöglichkeiten
- die weltweit meist benutzte Numerik-Software
- viele spezialisierte Toolboxen; jede Toolbox kostet extra
- Lizenz für Standardmodellierung: ca. € 10.000,- / Wartung: ca. € 1.800,- pro Jahr
- Lizenz für Codegenerierungstoolkette: ca. € 25.000,- / Wartung: ca. € 3.500,- pro Jahr

Professional VisSim der Firma Visual Solutions, USA :

- http://www.vissim.com/products/pro_vs/index.html
- Modellierung und Simulation komplexer, nichtlinearer Systeme
- blockdiagrammbasiertes Modellieren
- leistungsstarke Realzeitsimulierung und Anknüpfung an I/O-Karten für Datenerfassung
- findet Anwendung in der Luft- und Raumfahrtechnik, Automobilindustrie, Verkehrstechnik, Kommunikations- und Nachrichtentechnik
- insbesondere geeignet für kleine Systeme
- Lizenz: ca. USD 3.000,-

Saber der Firma Synopsys Inc., USA:

- http://www.synopsys.com/products/mixedsignal/saber/saber.html
- ähnlich Dymola
- objektorientiert, physikalisch, multidomän
- Proprietär
- Lizenz: ca. € 40.000,- / Wartung: ca. € 5.000,- pro Jahr

2.2 Simulationstools in der Antriebstrangsimulation

In der Automobilindustrie kommen für die verschiedenen Anwendungsgebiete in der Antriebstrangauslegung die unterschiedlichsten Simulations- und Modellierungstools zu Einsatz.
Häufig verwendet werden folgende drei Simulationstools:

ADVISOR der Firma AVL, Österreich:

- http://www.avl.com
- wurde entwickelt unter Leitung des U.S. Department of Energy, des National Renewable Energy Laboratory (NREL) der USA
- wird seit 2004 von der Fa. AVL, Österreich, vermarktet
- Tool für die Antriebstrangsimulation konventioneller sowie Hybrid- und Brennstoffzellenfahrzeuge
- basiert auf Matlab/Simulink
- enthält vorkonfigurierte Antriebstränge verschiedenster bekannter Hybridfahrzeuge
- dieses Tool kann individuell erweitert werden
- Lizenz/Wartung: ca. € 7.500,- pro Jahr

ASCET-SD Modeling & Design der Firma ETAS GmbH. Deutschland:

- http://de.etasgroup.com/products/ascet_sd/
- grafisches Modellierungstool
- hat Stärken eher bei der Steuergerätealgorithmenmodellierung und Steuergerätecodegenerierung und ist weniger stark in der Modellierung mechatronischer Systeme
- ein OSEK-konformes Betriebssystem wird in der Lizenz mitgeliefert
- Schnittstelle zu Matlab/Simulink ist vorhanden, die ständig verbessert wird
- Lizenz: ca. € 22.000,-

WAVE der Firma Ricardo Plc., GB:

- http://www.software.ricardo.com/products/wave/
- 1D-Thermodynamiksimulations- und modellierungstool
- blockdiagrammbasiertes Modellieren
- proprietäres Tool
- ca. € 20.000.-

Damit eine Antriebstrangmodellierung transparent ist und von verschiedenen Berechnern ohne massiven Einarbeitungsaufwand genutzt werden kann, wird an diese bzw. werden an ein Modellierungstool oder eine Modellierungsmethodik folgende Anforderungen gestellt:

Der Standard bzw. die in dem Tool verwendeten Basismethoden für die Modellierung von Komponenten sollen möglichst nicht-proprietär und offen sein. Das heißt es soll dem Anwender möglich sein, selber Änderungen an bestehenden oder auch käuflich erworbenen Modellen zu machen und neue Modelle zu erstellen. Daneben sollte der verwendete Modellierungsstandard allgemein anerkannt sein für die Modellierung mechatronischer Systeme. Des Weiteren sollte es möglich sein, die durch das Tool generierten Daten ohne Aufwand in andere Datenformate zu exportieren, damit diese Daten durch bereits vorhandene

Tools weiterverarbeitet werden können. Das heißt also, daß verwendete Datenformate und Schnittstellen zu anderen Tools z. Bsp. in Form einer API offengelegt sein sollten.

Da die eben genannten Anforderungen von keinem Tool alle zugleich erfüllt werden, wurden für die hier vorgestellte Gesamtfahrzeugsimulation mechatronischer Systeme die Vorteile von Matlab/Simulink und Dymola in der Weise genutzt, als daß Matlab/Simulink für die Gesamtintegration der hier vorliegenden Gesamtfahrzeugsimulation verwendet wird sowie für die Modellierung von diskreten Controlleralgorithmen und Dymola aufgrund seiner Objektorientiertheit und der Möglichkeit der mathematischen Beschreibung verschiedenster Domänen als Modellierungstool für die Physik, d.h. für die physikalische Regelstrecke Fahrzeug-Straße-Umwelt.

2.3 Unterschiede in der Modellierung zwischen Regelstrecke und Controller

Bei Regelungssystemen wird zwischen Regelstrecke und Regler unterscheiden. Will man diese in einem zu simulierenden Modell nachbilden, dann ist die Frage nach der Strukturierung der Modelle und der Art und Weise der Modellierung nicht trivial, da die verschiedenen Simulationswerkzeuge Vor- und Nachteile haben und möglicherweise Regelstrecke und Regler auf unterschiedlicher Art und Weise modelliert werden müssen. Speziell bei digitalen Reglern, wie sie heute sehr häufig in sogenannten Steuergeräten, den Controllern oder ECUs, realisiert werden, unterscheidet sich die Modellierungstechnik bei dem Entwurf von Algorithmen vom Modellieren physikalischer Regelstrecken.

So wird die Regelstrecke entsprechend der physikalischen Gegebenheiten möglichst objektorientiert strukturiert und auch modelliert.
Algorithmen zur Steuerung dagegen können verteilt auf verschiedene Controller implementiert sein. Aus diesem Grund sollte eine hierarchische, abstrakte Funktionsarchitektur existieren, wo definierte Schnittstellen festgelegt sind, die die Beziehungen zwischen den verschiedenen Ebenen und Objekten dieser Funktionsstruktur beschreiben. Diese Funktionsstruktur ist unabhängig von der tatsächlichen Verteilung der Algorithmen auf die Controller. Eine solche Funktionsstruktur wird zurzeit von verschiedenen Fahrzeugherstellern und Zulieferern, im Arbeitskreis Autosar (auch: AUT@SAR) entwickelt [Autosar 2004].

Abgesehen von der Art und Weise der Strukturierung der Modelle gibt es außerdem Unterschiede in der Art und Weise der Modellierung der Regelstrecke und von Controlleralgorithmen: Controlleralgorithmen werden Sinnvollerweise nur zeitdiskret simuliert, d.h. mit fester Schrittweite, wie das der Realität entspricht. Aufgrund der relativ groben Abtastung (üblicherweise im Millisekundenbereich) bedeutet dies auch keine Einschränkung hinsichtlich Rechnerbelastung. Daraus folgt, daß in Controllern keine Differentialgleichungen sondern nur Differenzengleichungen gelöst werden. Ein Spezialfall sind ereignisdiskrete oder sogenannte event-gesteuerte Unterfunktionen in Controllern, wie zum Beispiel die kurbelwellensynchrone Task im Motorsteuergerät. Auch hier werden Differenzen zum letzten Abtastschritt für das Lösen von Differenzengleichungen verwendet. Die Algorithmen sind bezüglich der Schrittweite zum letzten Abtastschritt diskretisiert.
Die Regelstrecke im Gegensatz dazu wird Vorteilhafterweise kontinuierlich, d.h. mit variabler Schrittweite simuliert. Durch die variable Integrationsschrittweite bzw. durch eine Schrittweitensteuerung wird abhängig vom Lösungsverfahren des verwendeten Solvers in vielen Situationen die Zeit des Konvergierens der Lösung wesentlich verkürzt im Vergleich zu Integrationsverfahren mit festen Schrittweiten.

Durch die unterschiedliche systemtheoretische Beschreibung der Übertragungsfunktionen in der Controllermodellierung und der Modellierung der Regelstrecke unterscheiden sich die in den Teilmodellen verwendeten Bibliotheksfunktionen (zum Beispiel Integration und Differentiation).

Controller werden in der Regel nur hinsichtlich ihrer eigentlichen Funktionssoftware in unterschiedlicher Detaillierungstiefe modelliert. So wird zum Beispiel das Umsetzen des CAN-Protokolls oder die stets umfangreiche Sicherheitssoftware im Allgemeinen nicht modelliert. (Eine Ausnahme in der vorliegenden Simulation ist der BMS-Controller: Hier wird als DLL fast die gesamte Steuergerätesoftware eingebunden, soweit diese unabhängig vom Zielsystem kompiliert werden kann. S. Kapitel 3.3)

Im Folgenden soll eine Übersicht der wichtigsten Unterschiede in der Modellierung von Regelstrecken und Controlleralgorithmen gegeben werden, die so auch Anwendung fanden in der Gesamtfahrzeugsimulation:

	Regelstrecke	**Controller**
Integrationsschrittweite	variabel	feste Schrittweite ΔT (Abtastrate bzw. Zeitdifferenz zum letzten Abtastschritt)
Verfahren zum Lösen von Differentialgleichungen	zum Beispiel implizites oder explizites Euler-Verfahren, Runge-Kutta usw.	Differenzengleichungen
Abtastrate $f_{ab} = 1/T_{ab}$	fest, entspricht der des schnellsten Controllers; es wird ein time-event ausgelöst	fest, für verschiedene Controller unterschiedlich
Beschreibung von zeitlich veränderlichen Zusammenhängen im Bildbereich (systemtheoretisch)	Laplace-Transformation für LTI-Systeme	z-Transformation
Beschreibung von zeitlich veränderlichen Zusammenhängen im Zeitbereich (systemtheoretisch)	analytisch (differential algebraische Gleichungen) ggf. unter Verwendung von Events zur Umschaltung des mathematischen Zusammenhanges	diskret
Verwendung von Bibliotheken (z. Bsp. in Simulink)	kontinuierlicher Blockset	diskreter Blockset
Strukturierung	objektorientiert, angelehnt an die Realität	eher funktionsorientiert
Schnittstellen zwischen internen Modellierungsobjekten	physikalische Signale, Gleitkommaarithmetik	Variablen, abhängig vom Aufbau des Prozessors integer- oder Gleitkommarithmetik
Inputs/Outputs (Schnittstellen nach außen)	analoge oder digitale Signale, ggf. Signale CAN-Bus-fähiger Sensoren	analoge oder digitale Signale, Signale der CAN-Botschaften

Tabelle 2.3.1: Unterschiede in der Modellierung einer physikalischen Regelstrecke und in der Modellierung eines Controlalgorithmus

2.4 Unterschiede zwischen kausaler und akausaler Modellierung

Bei dem Modellieren physikalischer Zusammenhänge können die betrachteten Systeme und Komponenten über kausale Methoden oder nicht-kausale, sogenannte akausale Methoden beschrieben werden.

Die kausale Beschreibung, wie zum Beispiel durch Blockdiagramme und Signalflüsse, setzt abhängig von der Dekomposition des Systems und der Komponentenebene direkte Zusammenhänge zu einer Vorgängergröße voraus, eine sogenannte a-priori Kausalität. Ein Output kann nur über algebraische Zusammenhänge und den entsprechenden Eingängen berechnet werden. Viele nichtlineare Systeme lassen sich durch die kausale Modellierung nur schwer darstellen, insbesondere dann, wenn die Anzahl der Zustandsgrößen zeitabhängig sind. Typische Vertreter der signalorientierten, kausalen Modellierung sind zum Beispiel die Modellierungen unter ASCET-SD oder Matlab/Simulink.

Bei der akausalen Modellierung gibt es keine expliziten Inputs oder Outputs auf einer bestimmten Komponentenebene. Komponenten werden anhand ihrer inneren physikalischen, mathematischen Zusammenhänge (algebraische oder Differentialgleichungen) und deren physischen Verbindungen zu anderen Komponenten beschrieben. Bei dem Verbinden dieser Komponenten miteinander werden Flüsse (zum Beispiel elektrischer Strom, hydraulischer Durchfluß oder magnetische Flüsse) und Potentiale (zum Beispiel elektrische, magnetische usw.) generiert. Das Modellierungstool transformiert das System der vorliegenden algebraischen und Differenzialgleichungen symbolisch in eine Form, die sich mit den bekannten Integrationsverfahren integrieren läßt. Vertreter der physikalischen bzw. akausalen Modellierung sind zum Beispiel die Modellierungen unter Dymola/Modelica oder Saber.

Unterschiede in der Modellierung ergeben sich aus den Möglichkeiten, die beide Methoden hinsichtlich der Modellierung dynamischer Systeme mit sich bringen und die im Tool selbst bzw. in der Historie der Toolentwicklung und der eigentlichen Intention des jeweiligen Tools begründet sind. In den folgenden Kapiteln soll die Modellierung durch Matlab/Simulink und Modelica/Dymola als typische Vertreter der eben genannten kausalen bzw. akausalen Modellierung anhand mehrerer Beispiele gegenübergestellt werden, da beide Tools so auch in dieser Clustersimulation zur Anwendung kamen.

Ein einfaches Beispiel für diesen Vergleich ist ein elektrischer Schwingkreis (Abbildung 2.4.1 (a)), der relativ einfach aufgebaut und damit anschaulich ist. Der Schwingkreis enthält Energiespeicher, die zeitlich veränderliche Zustandsgrößen zur Folge haben. Neben Differentialgleichungen aufgrund der Strom-/Spannungsbeziehungen an Kondensator und elektrischer Spule enthält solch ein Schwingkreis auch Nebenbedingungen aufgrund der Kirchhoffschen Gesetze. Somit stellt diese elektrische Schaltung ein einfaches, differential algebraisches Problem dar.

In einem zweiten Schritt soll aufbauend auf dem Modell der Ausgangsschaltung (Abbildung 2.4.1 (a)) diese Schaltung durch Hinzufügen eines weiteren parallelen Zweiges erweitert werden (Abbildung 2.4.1 (b)). Hier sollen die Möglichkeiten und Grenzen der jeweiligen Methoden und Tools in Bezug auf Wiederverwendbarkeit und Austauschbarkeit von Modellen aufgezeigt werden.

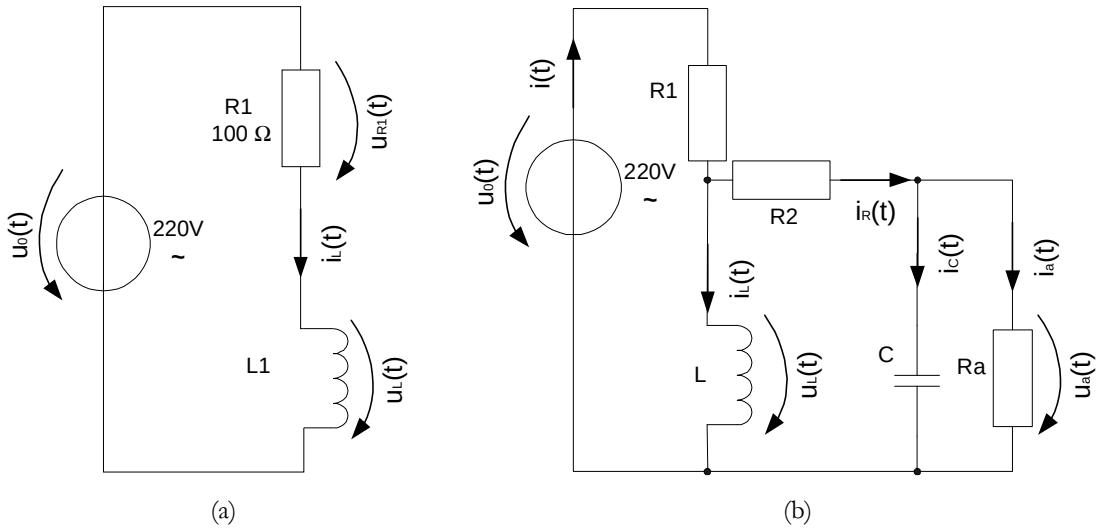


Abbildung 2.4.1: Elektrische Schaltung eines zu simulierenden elektrotechnischen Beispielsystems
(a) bestehend aus einer sinusförmigen Spannungsquelle parallel zu einem R-L-Zweig.
(b) Erweiterung der Schaltung aus Abbildung (a) um einen parallelen R-C-Zweig.

2.4.1 kausale (block- oder signalorientierte) Modellierung in Matlab/Simulink

Matlab wurde Anfang der 80-er des letzten Jahrhunderts von Cleve Moler und Jack Little in Natick, Massachusetts, USA, als Programm zum Lösen von Matrizengleichungen entwickelt und in der axiomatischen Systemtheorie für die Analyse von LTI-Systemen in Zustandsform angewendet (MATrix LABoratory für die Softwarepakete LINPACK und EISPACK). Simulink wurde später hinzugefügt, um mit Hilfe einer grafischen Oberfläche eine block- und signalorientierte Systemsynthese zu ermöglichen. Zuerst für Studenten entworfen wurde dieses Tool dann kommerziell verwertet.

Ursprünglich wurden Frequenzübertragungsfunktionen von LTI-Systemen analytisch untersucht. Das ist der Grund dafür, daß in Simulink, obwohl das Modell im Zeitbereich modelliert wird, im Blockschaltbild für die Integration eines kontinuierlichen Systems die Laplace-Übertragungsfunktionen 1/s dargestellt ist und sogar beliebige Laplace-Übertragungsfunktionen in das Modell eingefügt werden können. Eine Inkonsistenz im Sinne der Systemtheorie.
Simulink ist ein Simulations- und Modellierungstool in dem man blockorientiert Signalflüsse modelliert. Die Input-/Outputbeschreibung von aneinander geschalteten Blöcken muß hinsichtlich der Signalbeschreibung übereinstimmen, was die Austauschbarkeit von (Bibliotheks-) Elementen stark einschränkt.
Signale können Informationen über logische Größen und Zustandsgrößen weitergeben. So muß vom Anwender ein zu modellierendes System erst im systemtheoretischen Sinn analysiert werden, ehe für ein bestimmtes zu modellierendes Subsystem mit vorgegebenen Ein-/Ausgangsverhalten der Signalfluß aufgestellt werden kann.

Das eben Gesagte soll am Beispiel gemäß der Abbildung 2.4.1 (a) näher erläutert werden:
Es sei ein Teilmodell oder Subsystem zu entwerfen, wobei der Eingang der Schaltung die Spannungsquelle sein soll und der Ausgang die Spannung $u_{R1}(t)$ über dem Widerstand R1. Aufgrund des Modellierens in Signalflüssen ist die Spannung $u_{R1}(t)$ nicht über Potentiale definiert sondern ein Signal.

Aus $$u_0(t) = u_{R1}(t) + u_L(t) \qquad \text{(Gl. 2.1)}$$

folgt mit Hilfe der Gleichungen

$$u_L(t) = L \cdot \frac{di_L(t)}{dt} \qquad \text{(Gl. 2.2)}$$

und $$u_{R1}(t) = R_1 \cdot i_L(t) \qquad \text{(Gl. 2.3)}$$

$$u_{R1}(t) = u_0(t) - \frac{L}{R_1} \cdot \frac{du_{R1}(t)}{dt} \qquad \text{(Gl. 2.4)}$$

oder nach Integration

$$u_{R1}(t) = \frac{R_1}{L} \cdot \int (u_0(t) - u_{R1}(t))dt\,. \qquad \text{(Gl. 2.5)}$$

Es ist nicht klar, ob die Differentialform (Gl. 2.4) oder besser die Integralform (Gl. 2.5) für die (grafische) Beschreibung des in Abbildung 2.4.1 (a) dargestellten Zusammenhangs anwendet werden soll. Bezüglich des oben geforderten Ein-/Ausgangsverhaltens können die Strukturbilder in Simulink folgendermaßen aussehen (Abbildung 2.4.1.1):

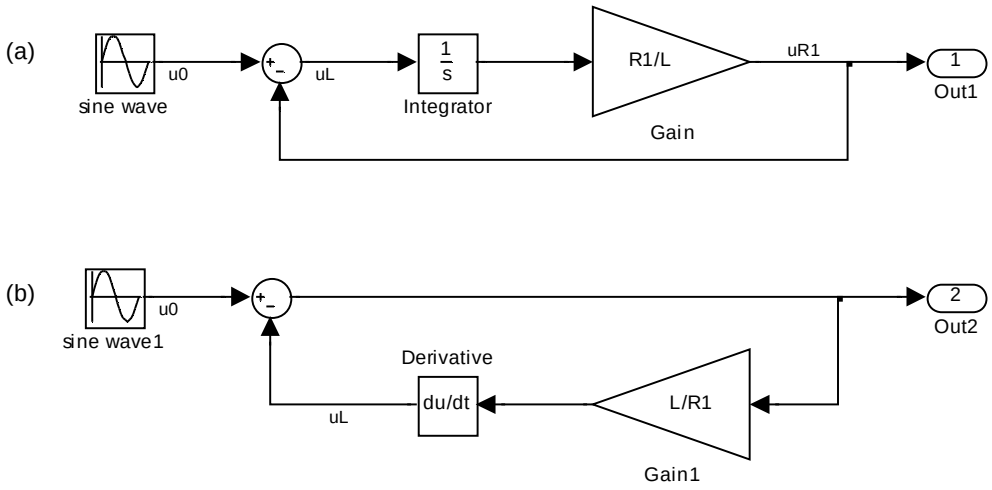


Abbildung 2.4.1.1: Strukturbilder in Simulink für ein und denselben Sachverhalt aus Abbildung 2.4.1 (a)
(a) gemäß Gl. 2.4 in Differentialform und
(b) gemäß Gl. 2.5 in Integralform.

Diese Blockschaltbilder spiegeln die physikalische Struktur der elektrischen Schaltung nicht mehr wider. Der Anwender muß sich mit Aspekten der Modellierung beschäftigen, wie das Aufstellen von Koppelbedingungen (z. Bsp. Erhaltungssätzen) und Übertragungsfunktionen und muß wählen, in welcher analytischen Form er den Sachverhalt darstellen will. Die Darstellung in Matlab/Simulink ist klar signalorientiert und die Energiespeicher, Zustandsgrößen und ihre Ableitungen sind erkennbar.

Nachfolgend sollen die Erweiterbarkeit, die Strukturierbarkeit und das Modellieren von nichtlinearen, zeitlich veränderlichen Simulinkmodellen untersucht werden.

1. Erweiterbarkeit von Modellen

Möchte man jedoch diese Schaltung gemäß der Abbildung 2.4.1 (b) erweitern, mit dem Ziel, die Spannung u_C am Ausgang zu messen, dann ist das Erstellen des Blockschaltbildes nicht mehr handhabbar, denn der Anwender selbst muß sich folgende Gedanken bezüglich der Herleitung der Zustandsgrößen u_L und i_L machen:

$$y(t) = u_C = i_a \cdot R_a \qquad \text{Ausgangsgleichung} \qquad \text{(Gl. 2.6)}$$

$$\dot{u}_C(t) = \frac{1}{C} \cdot (i_R - i_a) = \frac{1}{C} \cdot \left(\frac{u_L - y}{R_2} - \frac{y}{R_a} \right)$$
$$= \frac{1}{C} \cdot \left(\frac{u_L}{R_2} - \frac{u_C}{R_a} - \frac{u_C}{R_a} \right) \qquad \text{(Gl. 2.7)}$$

$$\dot{i}_L = \frac{1}{L} \cdot u_L \qquad \text{(Gl. 2.8)}$$

Gesucht wird $u_L = f(i_L, u_C)$

$$u_L = u_0 - i \cdot R_1 = u_0 - (i_L + i_R) R_1$$

$$= u_0 - i_L \cdot R_1 - (i_C + i_a) R_1 = u_0 - i_L \cdot R_1 - R_1 i_C - \frac{R_1}{R_a} u_C$$

in der Masche gilt: $u_L = (i_C + i_a) R_2 + u_C = i_C \cdot R_2 + \frac{R_2}{R_a} u_C + u_C$

$$i_C = \frac{u_L}{R_2} - \left(\frac{1}{R_a} + \frac{1}{R_2} \right) u_C$$

Somit ist $u_L = u_0 - i_L \cdot R_1 - \frac{R_1}{R_2} u_L + R_1 \left(\frac{1}{R_a} + \frac{1}{R_2} \right) u_C - \frac{R_1}{R_a} u_C$

und mit $R = R_1 + R_2$ folgt:

$$u_L = \frac{R_2}{R} u_0 - \frac{R_1 R_2}{R} i_L + \frac{R_1}{R} u_C \qquad \text{(Gl. 2.9)}$$

Damit folgt aus (Gl. 2.7) und (Gl. 2.9)

$$\dot{u}_C(t) = \frac{1}{CR} u_0 - \frac{R_1}{CR} i_L + \frac{1}{C} \left(\frac{R_1}{R_2 R} - \frac{1}{R_2} - \frac{1}{R_a} \right) u_C$$

$$= \frac{1}{CR} u_0 - \frac{R_1}{CR} i_L - \frac{1}{CR_a} \left(1 + \frac{R_a}{R} \right) u_C \qquad \text{(Gl. 2.10)}$$

und aus (Gl. 2.8) und (Gl. 2.9)

$$\dot{i}_L(t) = \frac{R_1}{LR} u_C - \frac{R_1 R_2}{LR} i_L + \frac{R_2}{LR} u_0 \qquad \text{(Gl. 2.11)}$$

Zum besseren Verständnis der im Folgenden vorgenommenen Substitutionen seien die Gleichung (Gl. 2.10) und (Gl. 2.11) in Zustandsnormalform notiert:

$$\begin{pmatrix} \dot{u}_C \\ \dot{i}_L \end{pmatrix} = \begin{bmatrix} -\frac{1}{CR_a} - \frac{1}{CR} & -\frac{R_1}{CR} \\ \frac{R_1}{LR} & -\frac{R_1 R_2}{LR} \end{bmatrix} \cdot \begin{pmatrix} u_C \\ i_L \end{pmatrix} + \begin{pmatrix} \frac{1}{CR} \\ \frac{R_2}{LR} \end{pmatrix} \cdot u_0 = \underline{\underline{A}} \cdot \begin{pmatrix} u_C \\ i_L \end{pmatrix} + \underline{b} \cdot u_0$$

Somit folgt unter Zuhilfenahme der Matrixelemente der Systemmatrix $\underline{\underline{A}}$ und des Vektors $\underline{b}$ das in der Abbildung 2.4.1.2 dargestellte Blockschaltbild mit dem eingangs geforderten Ein-/Ausgangsverhalten.

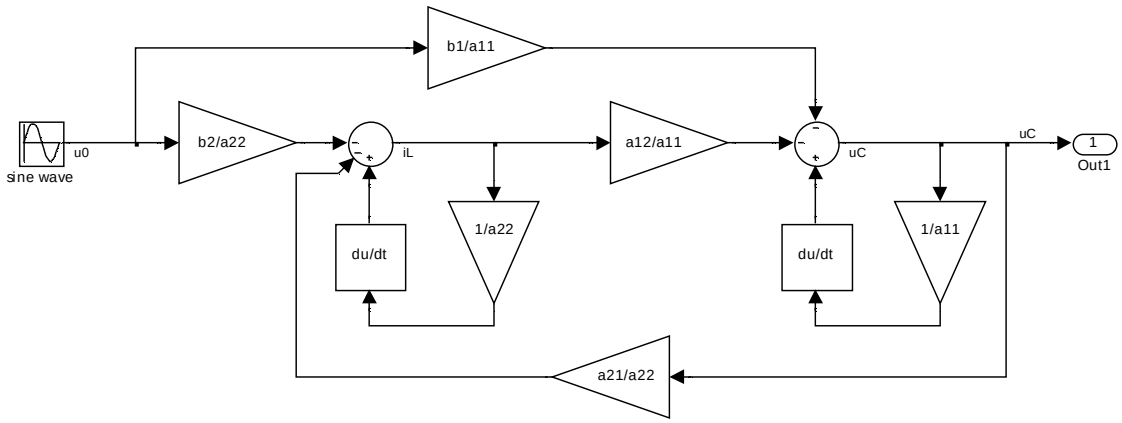


Abbildung 2.4.1.2: Das zum Schaltplan aus Abbildung 2.4.1 (b) äquivalente Blockschaltbild in Simulink. Modelliert wurde mit kontinuierlichen Blöcken der Simulink-Standardbibliothek.

Dieses Blockschaltbild ist nur eine Möglichkeit, die elektrische Schaltung gemäß der Abbildung 2.4.1 (b) zu modellieren. Der Aufwand, dieses Blockschaltbild zu entwickeln, war beträchtlich. Die physikalische Struktur ist verloren gegangen und kann am Modell nicht mehr abgelesen werden. Allerdings sind die Anzahl und die Kopplungen der Zustände erkennbar.

2. *Strukturierung von Modellen*

Sollen nun beide Schaltungsteile der Abbildung 2.4.1 (b) für sich modelliert werden, nämlich die ursprüngliche Schaltung gemäß der Abbildung 2.4.1 (a) und der erweiterte, parallele RC-Zweig, dann ist dies in getrennten Submodellen unmöglich, da Parameter beider Subsysteme in dieselben Verstärkerblöcke eingehen.
Auch am Beispiel eines einfachen Antriebstranges gemäß der Abbildung 2.4.1.3 läßt sich diese Schwierigkeit der signalorientierten Modellierung darstellen, wenn Teilmodellen getrennt entwickelt werden sollen.

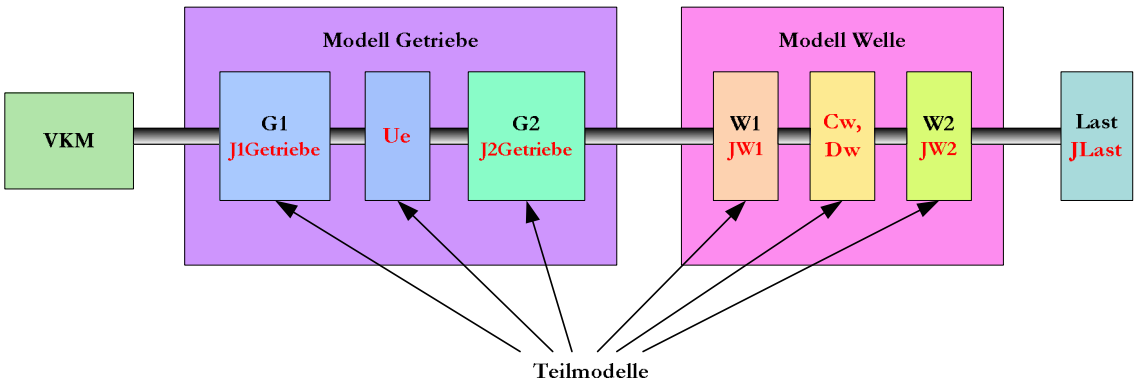


Abbildung 2.4.1.3: Einfaches Antriebstrangmodell mit den Teilwellen W1 und W2, den Trägheiten J1Getriebe, J2Getriebe, JW1, JW2 und JLast, der Dämpfungskonstanten Dw, der Federsteifigkeit Cw sowie dem Übersetzungsverhältnis Ue

Zwei Entwicklungsteams haben die Aufgabe, jeweils getrennt voneinander das Modell des Getriebes und der Welle zu entwickeln und zum Zweck der Wiederverwendung diese beiden Modelle sinnvoll in Teilmodelle zu strukturieren. Eine mögliche Lösung könnte die bereits in der Abbildung 2.4.1.3 dargestellte Systematik sein.

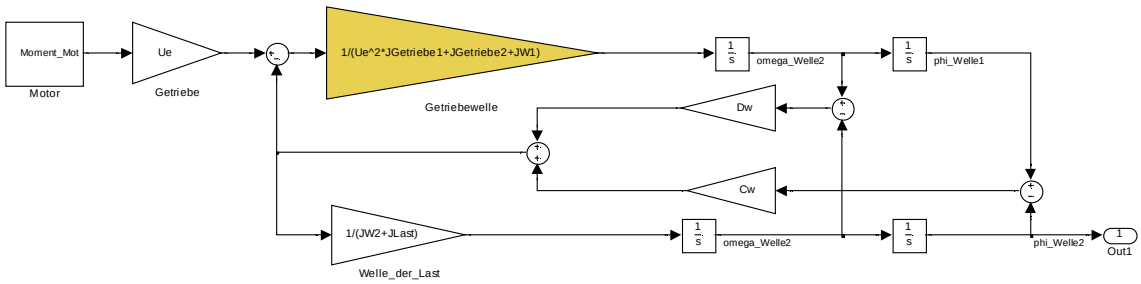


Abbildung 2.4.1.4: Simulinkmodell des Antriebstranges gemäß der Abbildung 2.4.1.3
Mit dem Präfix `omega_` sind die Wellenwinkelgeschwindigkeiten bezeichnet und mit `phi_` die Drehwinkel der Wellen.

Wie in Abbildung 2.4.1.4 ersichtlich, ist es nicht ohne weiteres möglich, die Modelle Getriebe und Welle des Beispielsystems in zwei unabhängigen Teams zu modellieren, da die Parameter der gekoppelten Teilsysteme G1, G2 und W1 in denselben Verstärkerblock eingehen. Nur durch Einführen von zusätzlichen Steifigkeiten z. Bsp. zwischen den trägen Massen JGetriebe2 und JW1 ließe sich das Gesamtmodell so strukturieren, daß es in zwei Teams modellierbar ist. Damit einher gehen die bekannten Probleme der Simulierbarkeit steifer Systeme (kleinere Simulationsschrittweite, möglicherweise Probleme in der Stabilität der Simulation usw.). Auch das Strukturieren des Modells Getriebe in seine Teilmodelle würde auf die gleiche, eben beschriebene Schwierigkeit stoßen.

Somit kann zusammenfassend festgestellt werden, daß die Dekomposition und damit das Strukturieren von Systemen in der kausaler Modellierung im Allgemeinfall nur mit Aufwand und dennoch nicht adäquat möglich ist. Dies bedeutet, daß bei kausaler Modellierung Objektorientierung im eigentlichen Sinn, d.h. das Halten von Subsystemen und Teilmodellen physikalischer Systeme in einer Modellbibliothek im Allgemeinen nicht möglich ist.

3. Modellieren von nichtlinearen, zeitlich veränderlichen Systemen

Die bis jetzt betrachteten Systeme waren linear und zeitlich invariant. Jedoch ist dies ein Spezialfall der einfach lösbaren Systeme. Der triviale Fall einer mechanischen Kupplung läßt sich mit der kausalen Modellierung nur schwer darstellen, da im offenen Fall zwei Trägheiten und mithin zwei Zustandsgrößen modelliert werden müssen und im geschlossenen Zustand der Kupplung nur eine Zustandsgröße. In diesem Fall handelt es sich um ein strukturvariables System. Nur durch Abstraktionen ist es möglich, diese signalorientiert zu Modellieren: Entweder man modelliert die Kupplung durch eine virtuell steife Kupplung zweier getrennter Zustandsgrößen oder durch eine explizite Umschaltung. Im ersten Fall kann es nötig sein, daß man die Simulationsschrittweite sehr klein wählen muß, um das steife System überhaupt simulieren zu können, was die gesamte Simulation unhandlich macht. Im zweiten Fall ist eine Art Zustandsautomat notwendig, der bewirkt, daß die explizite Umschaltung einen Momentensprung zur Folge haben kann.
Aufwendiger wird es, wenn Schaltgetriebe modelliert werden. Hier ändert sich die Systemübertragungsfunktion abhängig vom Schaltzustand. Schaltgetriebe sind ein Beispiel für

Systeme, die man besser nicht signalorientiert modelliert.
Nicht alle nichtlinearen Systeme sind ohne weiteres in Simulink simulierbar. Ist das differential algebraische System nicht in ein System von ODEs überführbar, dann kann dieses System in Simulink nicht gelöst werden. Das in der Abbildung 2.4.1.5 dargestellte Modell ist in Matlab/Simulink nicht modellierbar.

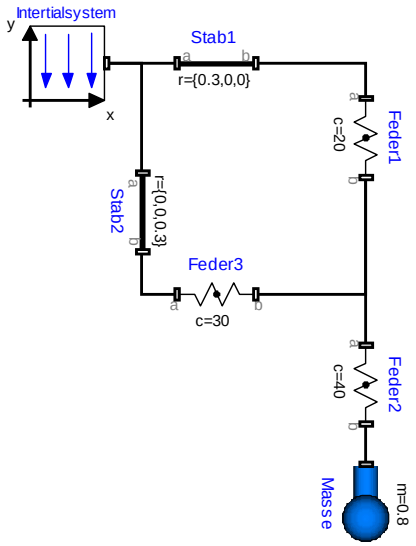


Abbildung 2.4.1.5: Modell eines Pendels, welches an drei miteinander verbundenen, masselosen Federn hängt. Die Stäbe sind im Raum mit einem Winkel ungleich 0 Grad zueinander angeordnet (z. Bsp. rechtwinklig zueinander).

Zusammenfassend läßt sich die kausale Modellierungsmethodik durch folgende Eigenschaften charakterisieren:

- Das zu beschreibende System wird in regelungstechnische Standardblöcke zerlegt, die das Systemverhalten der Regelstrecke erkennen lassen.
- Das Blockdiagramm entspricht einer Reihe von Anweisungen, die die Zustandsableitungen berechnen.
- Die Ordnung des Systems und die Kopplung der Zustände sind transparent.

Die blockorientierte Modellierung hat folgende Nachteile:

- Die Einsicht in die physikalische Struktur geht verloren.
- Die Blockdiagramme bzw. Strukturbilder sind schwer lesbar.
- Die Austauschbarkeit und damit Wiederverwendbarkeit von Teilmodellen wird erschwert.
- Eine kleine Änderung der physikalischen Struktur bedingt unter Umständen ein völliges Neuaufstellen des Blockdiagramms.
- Die Modellierung strukturvariabler, nichtlinearer Systeme ist entsprechend aufwendig und unübersichtlich (Diode, Kupplung, mechanischer Freilauf) und teilweise unmöglich.

Mehr zur Modellierung in Matlab/Simulink ist in [The MathWorks Inc. 2004] nachzulesen.

2.4.2 akausale (physikalisch orientierte) Modellierung in Modelica

Modelica ist ein offener, frei verfügbarer und nicht-proprietärer Modellierungsstandard, der Anfang der neunziger Jahre an der Universität Lund in Schweden in enger Zusammenarbeit mit der Deutschen Gesellschaft für Luft- und Raumfahrt (DLR) in Oberpfaffenhofen definiert wurde. [Otter et al. 2001, Modelica 2004]
Die grafische Oberfläche, die für die Modellierung nicht notwendig ist, wird durch Tools verschiedener Hersteller bereitgestellt. Unter anderem durch das kommerzielle Tool Dymola der Firma Dynasim AB, Lund in Schweden, welches große Verbreitung gefunden hat. Weiterhin gibt es das kommerzielle Tool MathModelica der Firma MathCore Engineering AB, Linköping in Schweden sowie die freie Entwicklungs- und Simulationsumgebung OpenModelica des Programming Environment Laboratory (PELAB) am Department of Computer and Information Science (IDA) der Universität Linköping in Schweden.

Aufgrund der ausgeprägten Möglichkeit der Multidomänmodellierung mit Hilfe Modelica werden insbesondere häufig mechatronische Systeme modelliert wie z. Bsp. Brennstoffzellensysteme, Flugsysteme, Klimaanlagen, Getriebe und Hybridfahrzeuge.

Wiederum wird die elektrische Schaltung aus Abbildung 2.4.1 (a) modelliert. Diesmal mit Hilfe von Standardelementen des Modellierungstools Dymola (Abbildung 2.1.2.1).

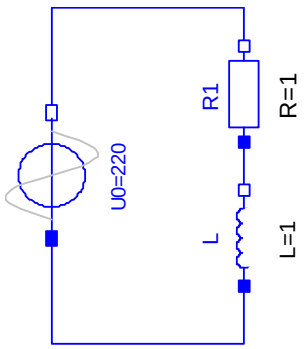


Abbildung 2.4.2.1: Die zum Schaltplan aus Abbildung 2.4.1 (a) äquivalente Modellierung in Dymola
Verwendet wurden Objekte aus der Standardlibrary von Dymola, die mathematisch in analytischer Form die Physik beschreiben.

Hierbei werden bereits vordefinierte Bauelemente mit ihren bekannten elektrischen Strom-/Spannungszusammenhängen symbolisch verschaltet. Jedes Element ist durch seine Schnittstelle an den Systemgrenzen (Potentiale und Flüsse) und sein inneres Verhalten mit Hilfe von Nebenbedingungen (z. Bsp. Erhaltungssätze, Knoten- oder Maschensätze) und Differentialgleichungen beschrieben.

Wie man sieht, entspricht das Modell auf dem ersten Blick der Struktur des Systems. Wenn dieses System erweitert wird gemäß der Abbildung 2.4.1 (b), bleibt die Lesbarkeit des Modells dennoch erhalten (Abbildung 2.4.2.2). Dabei ist zu beachten, daß dieses Erweitern nur ein Hinzufügen dreier Bauelemente ist, ohne daß irgendwelche Zusammenhänge neu erarbeitet werden müssen. Es existieren keine expliziten Eingangs- oder Ausgangsgrößen. Sämtliche realen, physikalischen und nichtphysikalischen Größen können hinsichtlich ihrer zeitlichen Entwicklung nach der Simulation ausgewertet werden.

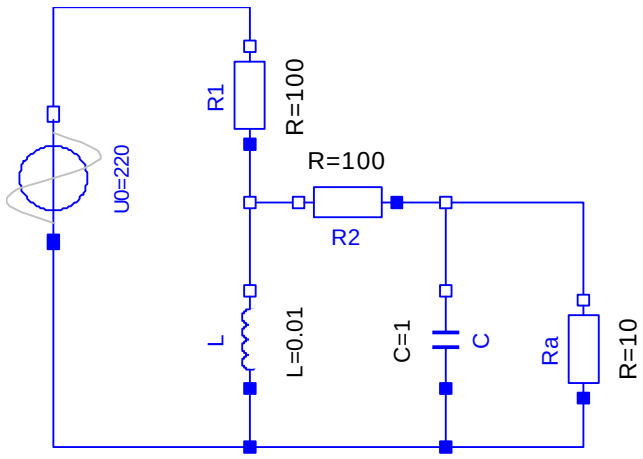


Abbildung 2.4.2.2: Dymolamodell gemäß dem Modell aus der Abbildung 2.4.1 (b)

Selbst bei gekoppelten Systemen erfolgt die Modellierung auf natürliche Weise durch die Komposition physikalischer Systeme: In der Abbildung 2.4.2.3 ist das Dymolamodell gemäß der Abbildung 2.4.1.3 dargestellt.

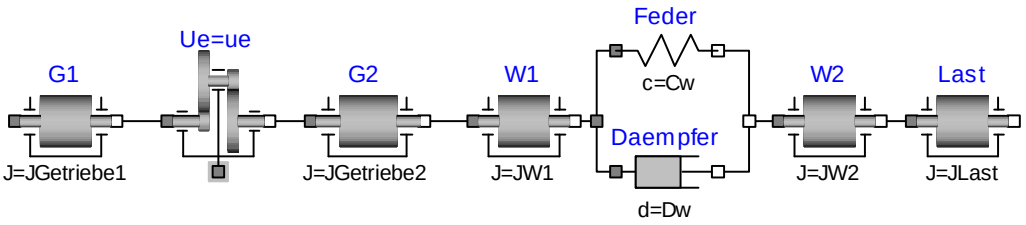


Abbildung 2.4.2.3: Dymolamodell gemäß dem Modell aus der Abbildung 2.4.1.3

Die gekoppelten Systeme bedeuten hier keine Schwierigkeiten, da Teilsysteme über Nebenbedingungen miteinander verbunden sind. So ist beispielsweise an der Verbindungsstelle zwischen den Teilmodellen G2 und W1 der Winkel ausgangsseitig G2 gleich dem Winkel eingangsseitig W1. Das System der differential algebraischen Gleichungen wird unter Berücksichtigung der Neben- bzw. Zwangsbedingungen aufgestellt, wenn möglich vereinfacht und symbolisch in eine für die Integrationsalgorithmen passende Form transformiert.

Selbst im Fall strukturvariabler Modelle (Kupplungen, Getriebe) können diese unter Zuhilfenahme von logischen Bedingungen, die die Art der Kopplung während der Simulation beeinflussen, modelliert werden.

Das Modell aus der Abbildung 2.4.1.5 ist mit dem Tool Dymola modelliert worden und läßt sich auf einfache Weise mit einem DAE-Solver simulieren.

Die akausale Modellierungsmethodik läßt sich durch folgende Eigenschaften charakterisieren:

- Das zu beschreibende System wird durch gekoppelte Standardbauteile, die gewöhnlich in Bibliotheken verfügbar sind, dargestellt.
- Die physikalische Struktur bleibt vollständig erhalten.
- Die Beschreibung entspricht den lokalen physikalischen Gleichungen der Bauteile, die von ihrer Umgebung unabhängig sind, sowie deren Kopplung zu einem Gesamtgleichungssystem.
- Das differential algebraische Gleichungssystem wird durch symbolische Transformation in eine für den Solver lösbare Form umgewandelt und numerisch integriert.

Die physikalisch orientierte Modellierung hat folgende Nachteile:

- Die regelungstechnische Streckenstruktur ist nicht erkennbar.

Mehr zur Modellierung in Modelica/Dymola ist in Kapitel 2.6 nachzulesen.

2.4.3 Bewertung der beiden Modellierungsansätze

Aus Modellierungssicht gibt es folgende Vorteile des akausalen Ansatzes: Da bei der akausalen Modellierung nicht signalorientiert modelliert wird, kann physikalisch strukturiert werden unabhängig vom kausalen Zusammenhang der Zustandsgrößen. Teilmodelle können da voneinander getrennt werden, wo es natürlich sinnvoll erscheint. Außerdem wird die Austauschbarkeit und Wiederverwendbarkeit von Komponenten und damit das verteilte Modellieren gewährleistet und die hierarchische Strukturierung des Modells unterstützt. Globale algorithmische Fallunterscheidungen sind bei strukturvariablen Systemen nicht notwendig. Das Modell ist in seiner physikalischen Struktur leicht änderbar. Es lassen sich physikalische Sachverhalte in Form von DAEs beschreiben, die sich häufig nur mit großem (händischen) Aufwand oder gar nicht in ein System von ODEs überführen lassen.

Aus regelungstechnischer Sicht scheint ein Vorteil des kausalen Ansatzes zu sein, daß die systemtheoretische Struktur der Regelstrecke erkennbar ist und damit sind die Ordnung des Systems. Die Kopplungen der Zustände untereinander sind transparent.
Der kausale Ansatz eignet sich eher für die algorithmische Beschreibung von Systemen, da Signalflüsse modelliert werden.

Eine Kombination der Vorteile beider Modellierungsansätze erscheint vorteilhaft und fand in der hier vorliegenden Gesamtfahrzeugsimulation Anwendung: Die Streckenmodellierung erfolgte nach dem physikalischen, akausalen Ansatz in Modelica durch das Tool Dymola. Das Tool Matlab/Simulink wurde für den Reglerentwurf in Blockdiagrammdarstellung verwendet. Die Simulation des Gesamtsystems findet in einer Co-Simulation zwischen den Simulatoren Dymola (variable step-size Solver) und Matlab/Simulink (fixed step-size Solver) statt.
Wenn es notwendig erscheint, den Matlab-Solver auch für die Regelstreckensimulation zu verwenden, dann kann das Dymolamodell durch Dymola in eine sogenannte Matlab/Simulink s-function konvertiert werden, indem das System der differential algebraischen Gleichungen in ein System von ODEs umgewandelt wird. (Wenn eine Umwandlung nicht möglich ist, dann wird eine Warnung angezeigt.) Im Fall der s-function wird das Dymolamodell auch mit einer festen Schrittweite gerechnet, die ein ganzzahliges Vielfaches der in Simulink eingestellten Schrittweite sein muß.
Als Spezialfall der Co-Simulation zwischen Matlab/Simulink und Dymola sei noch die Umwandlung des Dymolamodells in eine Matlab/Simulink s-function mit einem eigenen Solver erwähnt: Durch käuflichen Erwerb der RT-Option für Dymola (Echtzeit-Zusatzfunktion), ist es möglich, die zeitlichen Anforderungen, die an eine sogenannte harte Echtzeit gestellt werden, zu erfüllen. D.h., das Dymolamodell rechnet garantiert so schnell, daß das simulierte Zeitintervall der vergangenen Rechenzeit entspricht. Erreicht wird dieses Verhalten durch einen Dymola-eigenen fixed-step Solver.

Ein weiterer Vergleich der beiden Modellierungsansätze ist in [Tiller et al. 2003a] nachzulesen.

2.5 Modellierung in Modelica

Modelica wurde dafür geschaffen, sehr große, komplexe und heterogene Systeme modellieren zu können. So können derzeit Systeme mit über 300.000 Gleichungen und 300 Zustandsgrößen dargestellt und gelöst werden. Modelle in Modelica modelliert eignen sich für HIL-Simulationen und für die Entwicklung von Algorithmen für Embedded Controller Systemen, wofür sie auch eingesetzt werden.

Die Modelica Assocciation, eine nicht kommerzielle, nicht staatliche Organisation kümmert sich um die Weiterentwicklung, die Werbung und Anwendung des Modelica Programmierstandards.

Im Jahr 2000 fand die erste Internationale Modelica-Konferenz statt und 2002 die zweite, wobei dann die zweite, überarbeitete Version der Modelica-Sprachdefinition durch die Modelica Assocciation verabschiedet wurde. 2003 fand die dritte Internationale Modelica-Konferenz statt. Bis 2003 wurden über 700 kommerziell genutzte Lizenzen verkauft. Die Vortragenden bzw. Anwender von Modelica arbeiten u. a. auf dem Gebiet der Luftfahrt, Automobilindustrie [Soejima 2000] und Robotik. Anfang 2005 wurde die Version 2.2 des Modellierungsstandards Modelica durch die Modelica Association verabschiedet. [Modelica 2005].

Typische Anwendungsgebiete für Modelica sind die Modellierung und Simulation

- von Brennstoffzellen z. Bsp. durch Firmen wie United Technologies, Inc., Daimler Chrysler AG oder der Universität Lund in Schweden,
- von konventionellen und hybridischen Antriebssystemen z. Bsp. durch Firmen wie Ford Company, Inc., Toyota oder diverser Universitäten,
- von konventionellen und Automatikgetrieben z. Bsp. durch Firmen wie Ford Company, Inc., BMW AG, ZF Friedrichshafen,
- von Elektromaschinen und Elektroniksystemen durch verschiedene Universitäten in Deutschland, USA, Italien und Schweden,
- von Heiz- und Kühlsystemen und thermodynamischen Systemen z. Bsp. durch Firmen wie Daimler Chrysler AG oder der DLR,
- von Robotiksystemen (z. Bsp. DLR, KUKA) sowie
- von Systemen in der Luft- und Raumfahrt (z. Bsp. Simulation des Flugzeugs A300-600ST der Firma Airbus S.A.S., bekannt unter dem Namen BELUGA).

Siehe dazu auch [Otter et al. 2002, Fritzson et al. 2003] sowie die Konferenzbände der verschiedenen internationalen Modelica Konferenzen [Modelica 2004].

Die Vorteile von Modelica gegenüber anderen Beschreibungssprachen sind neben dem günstigen Preis die Tatsachen, daß Modelica nicht auf eine Domäne beschränkt ist, daß es ein offener Standard zur objektorientierten Beschreibung ist und daß sich durch Modelica beschriebene Modelle einfach wiederverwenden, austauschen und auf andere Rechnersysteme portieren lassen, da die Beschreibung textuell erfolgt. Der Syntax von Modelica ist ähnlich wie bei FORTRAN oder ASCL der mathematischen Beschreibung entlehnt und leicht lesbar.

Modelica unterstützt beides: Modellieren auf einer abstrakten Ebene über die Komposition von Teilmodellen sowie detailliertes, gleichungsbasiertes Modellieren von Bibliothekskomponenten. Modelle bzw. physikalische Zusammenhänge werden in Modelica als algebraische oder diskrete Gleichungen sowie als Differentialgleichungen beschrieben. Es müssen keine Gleichungen per Hand gelöst werden.

Es gibt nur ein Basiselement und das ist das Objekt *class*. Alle anderen Objekte sind davon abgeleitet. Physikalische Systeme sind hierarchisch aus Objekten aufgebaut, die vorher definiert wurden. Objekte aller Hierarchieebenen können in Bibliotheken, sogenannten Packages, angelegt, instanziiert und vererbt werden. Die Objekte werden nicht, wie in Matlab/Simulink, signalorientiert beschrieben sondern grundsätzlich mathematisch aufgrund ihrer Potentiale und Flüsse. Die Interfaces beschreiben die Anbindungen nach außen.

Modelica wurde mit dem Ziel entworfen, daß Algorithmen für die symbolische Transformation auf das System der Modellgleichungen angewendet werden können, um dieses System der differential algebraischen Gleichungen symbolisch umzuwandeln und ggf. zu reduzieren und es anschließend mit Standardmethoden zu integrieren. Mit speziellen Algorithmen können auch gemischte Systeme, in denen neben reellen Unbekannte auch boolesche Unbekannte vorkommen, gelöst werden.

Dymola als ein kommerzielles Entwicklungswerkzeug, welches den Modelica-Standard zur Modellbeschreibung anwendet, besteht aus einem Modellübersetzer, der alle notwendigen symbolischen Transformationen für sehr große Modelle durchführen kann sowie aus einen grafischen Editor und bietet Interfaces unter anderem zu den Simulationswerkzeugen ASCET-SD, Saber und Matlab/Simulink an.

Als weiterführende Literatur zum Thema Modelica-/Dymolamodellierung ist [Elmqvist et al. 2002, Fritzson 2004] zu empfehlen.

2.6 Konzept der Gesamtfahrzeugsimulation

Bei der Auswahl des Konzeptes der vorliegenden Gesamtfahrzeugsimulation wurde den Belangen der Automobilindustrie und insbesondere der Konzernforschung der Volkswagen AG Rechnung getragen. Vor diesem Hintergrund ist die folgende Bewertung bzw. Entscheidung bei den einzelnen Kriterien zu verstehen.

In den folgenden Unterpunkten wird anhand von Kriterien wie zum Beispiel Handhabbarkeit die Auswahl der verwendeten Simulatoren beschrieben. Zuletzt wird der prinzipielle Aufbau der hier vorliegenden Gesamtfahrzeugsimulation erläutert.

1. Handhabbarkeit, Verbreitung

Eine Anforderung an ein Modellierungs- oder Programmiertool ist neben einer großen Verbreitung bzw. hohen Akzeptanz im Hinblick auf den betrachteten Gegenstand der Simulation (z. Bsp. Eignung für Multidomänmodellierung) auch die einfache Bedienbarkeit. In der jüngsten Vergangenheit hat sich speziell auf dem Gebiet der Bedienbarkeit die grafische Modellbildung als Stand der Technik in der Regelstreckensimulation herauskristallisiert. Zumindest als Darstellungsmethode für die Dekomposition von Modellen eignet sich die grafische Modellbildung hervorragend. Die Übersichtlichkeit von Modellen und damit die Wartbarkeit werden erhöht. Änderungen in der Modellstruktur sollen ohne hohen Aufwand möglich sein. Dies und damit auch die Austauschbarkeit werden durch objektorientierte Simulatoren unterstützt.

In der Automobilindustrie wurden in den letzten 20 Jahren Steuergerätealgorithmen in der Programmiersprache C oder teilweise auch in Assembler entwickelt. Die hier vorliegende Gesamtfahrzeugsimulation soll das Einbinden von Steuergerätecode als C-Code oder in Form einer vorkompilierten DLL unterstützen. Dies ist gegeben, wenn Matlab/Simulink als Tool für die Modellierung der Controller gewählt wird, da in Simulink C-Code in Form einer DLL als s-function eingebunden werden kann.

Gegenwärtig werden auch in der Serienentwicklung von Steuergerätecodealgorithmen für Kraftfahrzeuge immer häufiger Teilfunktionalitäten prototypisch mit Hilfe grafischer Modellierungstools wie zum Beispiel ASCET-SD oder Matlab/Simulink umgesetzt. Sogenannte automatische Codegenerierungen funktionieren heute schon sehr gut, so daß die Spezifikation bzw. Modellierung dieser Funktionen grafisch erfolgen kann, was Vorteile in der Übersichtlichkeit haben kann gegenüber der prozeduralen Programmierung wie zum Beispiel in der Programmiersprache C.

Aus diesem Grund und wegen der Verbreitung von Matlab/Simulink in der Automobilindustrie wurde Matlab/Simulink für die Modellierung der Controller und auch als Integrationstool für die gesamte Simulation gewählt.

Für die Modellierung der Regelstrecke wurde Modelica in Verbindung mit dem Programmiertool Dymola gewählt. Hintergrund für die Entscheidung für Dymola/Modelica war die Möglichkeit der Multidomänmodellierung und der mathematischen Beschreibung der Physik sowie die Objektorientiertheit, was ein großer Vorteil ist, wenn man verschiedene Antriebstrangkonfigurationen untersuchen möchte und schnell auf Änderungen reagieren muß.

2. Verfügbarkeit, Kosten

Die zu favorisierenden Tools sollten verfügbar sein und keine allzu hohen Kosten verursachen. Vor allem wegen der Kosten wurde bei der Controllermodellierung Matlab/Simulink dem ASCET-SD vorgezogen bzw. bei der Modellierung der Regelstrecke das Dymola dem Saber. Des Weiteren stehen auf dem Lizenzserver der Volkswagen Konzernforschung eine Anzahl von Matlab-Lizenzen zur freien Verfügung.

3. Simulationsschrittweite, Simulierbarkeit

Da die vorliegende Gesamtfahrzeugsimulation Modelle von verschiedenen Controllern enthält, kann es keine einheitliche Simulationsschrittweite geben. Auch ist es nicht sinnvoll, das gesamte Modell mit nur einer festen Schrittweite, nämlich der des am feinsten abgetasteten Teilmodells, zu simulieren. Vielmehr ist es zielführend, die für jedes Teilmodell sinnvolle Simulationsschrittweite zu wählen. Durch die Simulatorkopplung in der vorliegenden Gesamtfahrzeugsimulation ist es möglich, die Controller mit unterschiedlichen Schrittweiten zu simulieren. Diese Teilmodelle werden entsprechend der Abtastung ihrer Schnittstellendaten miteinander synchronisiert – rechnen aber ansonsten autonom. Das physikalische Modell wird mit variabler Schrittweite simuliert und die einzelnen Controller mit einer festen, der Realität entsprechenden Simulationsschrittweite.

Ziel der Gesamtfahrzeugsimulation ist eine 20-fache Echtzeit.

4. Methode der Toolkopplung und zeitliche Steuerung von Simulationen

Die verteilten Simulationen werden parallel abgearbeitet und zentral synchronisiert. Dabei spielt die echte Zeit keine Rolle. Für die Synchronisation wird eine zentrale Simulationszeit verwendet.

Die Kopplung verschiedener Tools bei einer komplexen Simulation hängt eng mit der zeitlichen Steuerung der Simulation zusammen. Folgenden Methoden der Toolkopplung können unterschieden werden:

- **Co-Simulation:** Die Simulationen der unterschiedlichen Simulatoren, d.h. Simulationswerkzeuge, laufen als verteilte Simulationen auf mehreren Rechnern parallel ab bzw. quasiparallel bei Simulationen auf einem Rechner.

- **Modelltransformation:** in das jeweils andere Simulatorumfeld: Wenn aus bestimmten Gründen eine Co-Simulation unterschiedlicher Simulatoren nicht möglich ist, dann hilft die Methode der Modelltransformation weiter. Als Beispiel sei das Einbinden von DLLs oder sogenannten s-functions in Matlab/Simulink genannt: Ein physikalisches Modell in Modelica, aus dem mit Hilfe von Dymola eine s-function generiert wird, kann unter Verwendung dieser s-function in ein Matlab/Simulinkmodell eingebunden werden. Bei dieser Art der Simulatorkopplung ist keine verteilte Simulation möglich.

- **Mischsimulation:** Selbstverständlich können die beiden eben genannten Methoden der Simulatorkopplung in einer komplexen Simulation nebeneinander Anwendung finden.

Darüber hinaus können folgende Simulationen unterschieden werden:
(entnommen [Miegler et al. 2002])

- ***„Lokale Simulation:*** *Der unkomplizierteste und schnellste Weg besteht darin, das komplette Verbundmodell innerhalb eines Simulators (z. B. Simulink) aufzubauen und auf einem Rechner zu simulieren.*
  Vorteil: Diese Methode ist für kleinere Modelle gut geeignet und erfordert – vom Simulator abgesehen – keine zusätzliche Software und Hardware.
  Nachteil: Unserer Erfahrung nach wird selbst bei mäßig komplexen Verbundmodellen schnell die Leistungsgrenze handelsüblicher PCs erreicht. Mit zunehmender Verfeinerung wird die Rechenzeit für das Modell so hoch, daß ein effektives Testen nicht mehr möglich ist. Man kann die CPU-Belastung verringern, wenn man das Modell mit Hilfe eines Code-Generators in ein C-Modell umwandelt. Aber auch hier sind die Grenzen schnell erreicht. Eine weitere Möglichkeit besteht darin, das Modell auf einen dedizierten Echtzeitrechner zu portieren. Dort wird das Modell sehr effizient simuliert. Die Anschaffungskosten für Echtzeitrechner sind jedoch hoch.
  Fazit: Aufgrund der begrenzten Rechenleistung eines Einzelrechners ist diese Methode nur für Modulentwicklung und kleine Systeme anwendbar.

- ***Synchrone Simulation auf verteilten Rechnern:*** *Bei der Synchronsimulation werden mehrere Simulatoren auf verschiedenen Rechnern miteinander gekoppelt. Man ist hierbei nicht notwendigerweise an einen bestimmten Simulator gebunden. Des Weiteren kann die Rechenleistung jederzeit durch Hinzunahme eines weiteren Rechners an den jeweiligen Bedarf angepaßt werden. …*
  Vorteil: Da alle Simulatoren synchron laufen, kann die Simulation sowohl in Zeitlupe als auch im Zeitraffer durchgeführt werden. Somit lassen sich beispielsweise schnelle transiente Effekte im Detail studieren. Selbst ein komplettes Anhalten der virtuellen Zeit ist möglich, um z. B. den C-Code eines Steuergeräts zu debuggen, ohne daß das Restsystem währenddessen seinen Zustand ändert. Massensimulation (Fehlersimulationen, Streusimulation) können im Zeitraffer durchgefahren werden. Auch Echtzeit ist mit dieser Methode prinzipiell möglich, jedoch nicht erforderlich, da durch die Synchronisation die virtuelle Zeit für alle Teilnehmer gleich ist. Somit werden alle erforderlichen Zykluszeiten oder Haltezeiten mit den gleichen Zeitfaktor skaliert. Auch Schwankungen in der Rechenzeit sind damit kein Problem.
  Nachteil: Damit ein Verbund aus gekoppelten Simulatoren reibungslos funktioniert, ist ein aufwendiges Übertragungsprotokoll erforderlich. Das Protokoll ist in mehreren Schichten aufgebaut, die die sichere Übertragung der Daten gewährleisten, und enthält neben den Signaldaten protokollbedingte Zusatzdaten, die einen nicht unerheblichen Overhead erzeugen. Folglich muß man bzgl. der Partitionierung des Datennetzwerks und der Verteilung der Modelle sorgfältig planen, damit der Rechenleistungsgewinn nicht durch ein hohes Datenübertragungsaufkommen zunichte gemacht wird. …
  Fazit: Die Methode ist anwendbar von der Modulentwicklung bis zur Systemintegration. Die Möglichkeit, die virtuelle Zeit zu steuern (Zeitlupe und Zeitraffer) erleichtert die detaillierte Fehlersuche und Fehlersimulationen. Die Einbindung von Hardware unter Echtzeitbedingungen ist nur eingeschränkt möglich. Es können verschiedene Simulatoren eingesetzt werden.

- ***Asynchrone Simulation auf verteilten Echtzeitrechnern:*** *Es wird ein Verbund aus asynchron arbeitenden Rechnern aufgebaut, die alle in Echtzeit simulieren. Da die simulierte Zeit somit der realen Zeit entspricht, ist eine zeitliche Synchronisation der Simulatoren nicht erforderlich. Die Anforderungen an das Datenaustauschprotokoll sind wesentlich geringer, da keine Zeitstempel und sonstige simulatorspezifische Steuerdaten übertragen werden müssen. Für den Simulinksimulator sind mehrere solche Echtzeitsysteme am Markt verfügbar (dSPACE, xPC-Target). Die Datenvernetzung erfolgt durch ein geeignetes Bussystem (TCP/IP, CAN etc.).*

Vorteil: Da alle Rechner in Echtzeit laufen, ist die Anbindung von Hardware (Hardware-in-the-Loop) problemlos möglich. Die Mischung von echten und simulierten Komponenten im Verbund kann in nahezu beliebiger Weise erfolgen. Da alle Rechner asynchron arbeiten, unterliegt die Auswahl der Rechnerhardware, des Betriebssystems und des Simulators keinen Einschränkungen. Einzige Voraussetzung ist, daß alle Rechner das gleiche Netzwerkprotokoll unterstützen. Der CAN-Bus ist beispielsweise eine gute Wahl, da derzeit viele im automobilen Umfeld verwendeten Echtzeitrechnersysteme diesen Bus unterstützen.
Nachteil: Der asynchrone Rechnerverbund kann nur in Echtzeit arbeiten. Zeitraffer und Zeitlupe sind nicht mehr möglich, da keinerlei Synchronisation erfolgt. Des weiteren müssen die verwendeten Simulatoren harten Echtzeitbedingungen genügen, was die Flexibilität der Simulatoren und Modelle einschränkt. Die verwendeten Modelle müssen bei fest vorgegebener Zeitschrittweite stabil arbeiten. Iterative Verfahren zur Fehlerkorrektur und Zeitschrittsteuerung, die von Simulatoren zur Verbesserung der Modellkonvergenz eingesetzt werden, sind nicht zulässig, da diese keine festen Abarbeitungszeiten unter Echtzeitbedingungen garantieren können. Um diesen Vorgaben gerecht zu werden, wird aus dem Simulationsmodell meist ein sog. Echtzeit-Target generiert. Hierbei handelt es sich um ein C-Modell, daß ein stark vereinfachtes Lösungsverfahren enthält und für das jeweilige Zielbetriebssystem optimiert ist. Da während der Simulation der eigentliche Simulator nicht mehr zu Verfügung steht, sondern ein vereinfachtes C-Modell, ist der Benutzer nicht mehr in der Lage, in seiner gewohnten Entwicklungsumgebung funktionelle Änderungen am Modell vorzunehmen. Auch Debuggen und Tracen von Modellen sind nicht mehr möglich, da die simulierte Zeit nicht angehalten werden kann.
Fazit: Diese Methode ist in den späten Entwicklungsphasen zu empfehlen, wenn nur noch wenige funktionale Änderungen an den Modellen vorgenommen werden. Die Ankopplung von ist Hardware möglich. Die Simulation läuft ausschließlich in Echtzeit."

Weiteres zu diesem Thema kann [Kovacevic 2002] entnommen werden.

Bei der in dieser Arbeit vorgestellten Gesamtfahrzeugsimulation handelt es sich um eine räumlich verteilte, zeitlich synchrone Co-Simulation. Gesteuert wird diese Simulation zentral durch ein Mastermodell.

Für die Kopplung der physikalischen Regelstrecke mit den Controllern, die als Simulinkmodelle oder als DLL vorliegen, wurde die Simulatorkopplung der Firma Extessy AG aus Wolfsburg verwendet und weiterentwickelt [EXITE 2004]. Hierbei wird von jedem Teilmodell eine Input-/Outputinterfacebeschreibung, das sogenannte Klasseninterface, erstellt. Die Gesamtfahrzeugsimulation wurde so konfiguriert, daß auf einem Masterrechner die Kommunikation zwischen allen Modellteilen erfolgt, die auf den verteilten Slaves abgearbeitet werden (s. Kapitel 4.2 und 4.3). D.h. der Masterrechner und somit das Mastermodell haben Verbindungen zu allen Teilmodellen. Das Produkt der Firma Extessy, welches diese Dienste bereitstellt, heißt EXITE. Für verschiedene Simulatoren wie Matlab/Simulink, Saber oder ASCET-SD gab es bereits eine EXITE-Anbindung. Für das Simulationswerkzeug Dymola, aus eigenen C-Code generierte ausführbare Modelle sowie die aus Matlab/Simulink über den Real-Time Workshop (RTW) generierten ausführbaren Dateien (sog. executables) wurde die EXITE-Anbindung im Rahmen der Gesamtfahrzeugsimulation zusammen mit der Firma Extessy erarbeitet.

5. *Methode der Momentenberechnung*

In der Antriebstrangsimulation rückgekoppelter dynamischer Systeme unterscheidet man zwei Methoden: die Vorwärtsimulation und die Rückwärtssimulation.

Bei der Rückwärtssimulation werden ausgehend von einer bestimmten Fahrzeuggeschwindigkeit die notwendigen Radmomente berechnet und über Achs- und Getriebeübersetzungen die notwendigen Momente des Elektromotors bzw. Verbrennungsmotors.

Bei der Vorwärtssimulation werden ausgehend von einer Zyklusvorgabe und einem Fahrermodell, in welchem der Soll-/Istvergleich der Fahrzeuggeschwindigkeit erfolgt und in welchem die Gas- und- Bremspedalstellungen berechnet werden, die Momente von Verbrennungskraftmaschine und Elektromotor gebildet. Dem Antriebstrang bis zu den Rädern folgend werden diese Momente über die Getriebe- und Achsübersetzung in Radantriebs- bzw. -bremsmomente umgerechnet, so daß sich den Fahrwiderständen entsprechend eine Fahrzeugistgeschwindigkeit ergibt [Lin et al. 2001, Kube et al. 2004].

In der vorliegenden Simulation hybridischer Antriebstränge wurde die Vorwärtssimulation angewendet.

6. *Prinzipieller Aufbau der Clustersimulation*

Die Regelstrecke wurde in Dymola/Modelica modelliert und die Controlleralgorithmen blockorientiert in Matlab/Simulink bzw. als DLL eingebunden. Die Gesamtintegration durch das Mastermodell erfolgte in Matlab/Simulink (Abbildung 2.6.1).

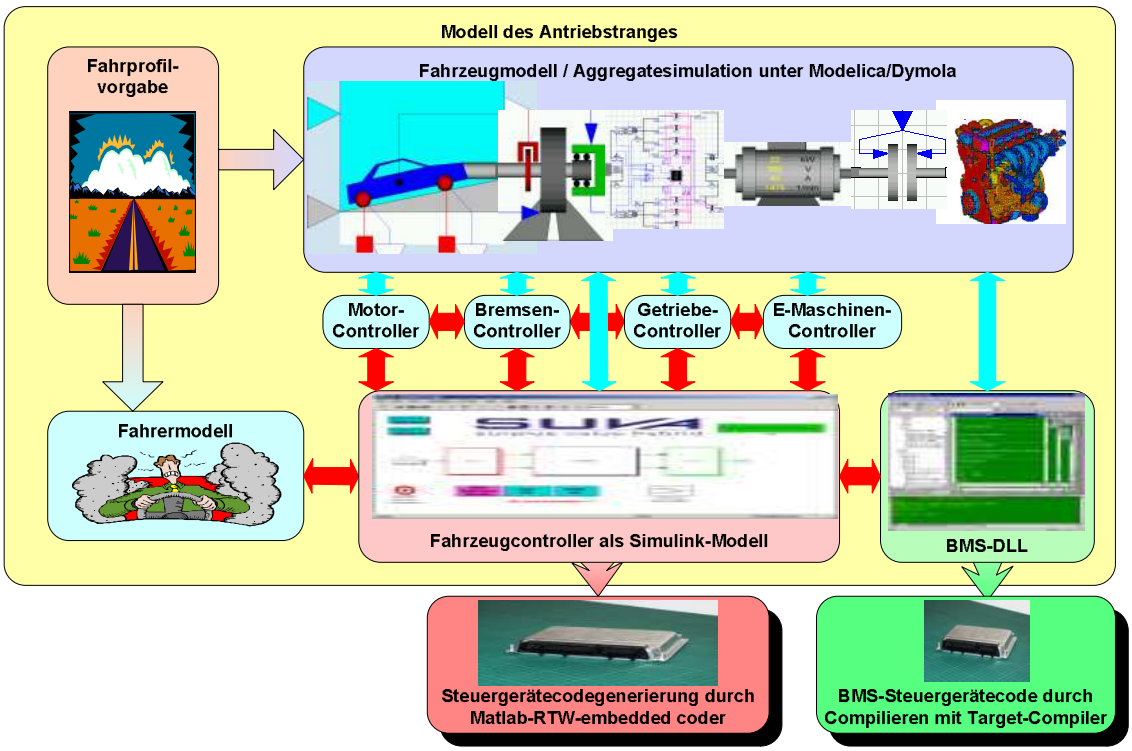


Abbildung 2.6.1: Übersicht über den modelltechnischen Aufbau der Gesamtfahrzeugsimulation. Die Pfeile bezeichnen den Austausch von Daten zwischen den Teilmodellen in Form von CAN-Signalen (rot) und physikalischen Signalen (Zyan). Ausgangswerte der Fahrprofilvorgabe sind Zeitverläufe physikalischer Signale wie zum Beispiel Fahrzeuggeschwindigkeit, Luftdruck oder Steigung.

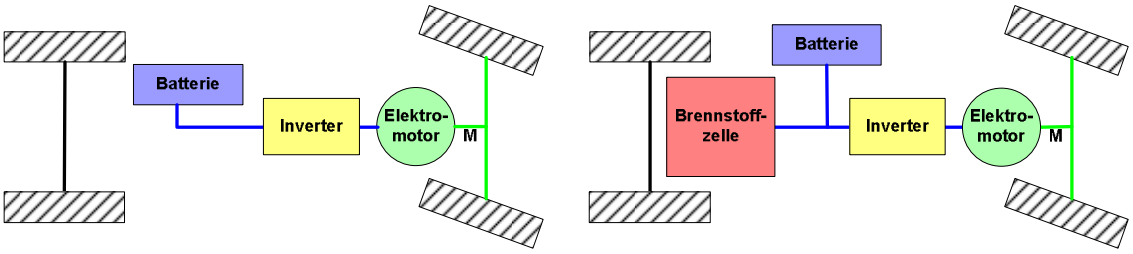
Der CAN-Bus wird auf Signalebene simuliert, d.h. zwischen den Teilmodellen werden die Signale gemäß der CAN-Datenbank in 64 Bit große Datenpakete gepackt (8 x 8 Bit), mit einer Abtastzeit als konstante Verzögerungszeit übertragen und im Empfängermodell entpackt. Dies spiegelt ein ideales Verhalten des CAN wider. Das Verhalten des CAN-Busses, d.h. die physikalische Beeinflussung durch zum Beispiel elektromagnetische, nichtleitungsgebundene Störungen, die realen und nicht konstanten Totzeiten, d.h. das Jittern der Botschaften sowie das Verhalten der Sicherheitsschicht einschließlich der Checksummenbildung wird nicht simuliert.

Das Ergebnis der hier vorliegenden Gesamtfahrzeugsimulation ist neben einem tieferen Verständnis für die Funktionsweise der einzelnen Aggregate und des Gesamtsystems eine Spezifikation des Steuergerätecodes für einzelne Steuergeräte, wie zum Beispiel für das Energiemanagementsteuergerät VMU oder das Batteriemanagementsteuergerät BMS. Für die beiden eben genannten Steuergeräte kann im Anschluß an die Simulation durch den Prozeß der automatischen Codegenerierung im Fall der VMU bzw. durch das Kompilieren des Codes mit dem Targetcompiler im Fall des BMS downloadfähiger Steuergerätecode generiert werden.

3 Modellbildung

Die Volkswagen Konzernforschung beschäftigt sich seit über 30 Jahren mit der Hybridisierung von Antriebssträngen von Kraftfahrzeugen bzw. mit alternativen Antriebskonzepten [Josefowitz et al. 2002].

Konzepte für alternative Antriebstränge sind zum Beispiel die reinen Elektrofahrzeuge aber auch die sogenannten Brennstoffzellenfahrzeuge. Diese Fahrzeuge besitzen keinen Verbrennungsmotor mehr. Ein Elektromotor treibt diese Fahrzeuge an und erzeugt das notwendige Vortriebsmoment. Durch Rekuperation, d.h. Rückspeisung der Energie durch Betreiben des Elektromotors als Generator, erfolgt das sogenannte regenerative Bremsen, solange damit die gewünschte Bremswirkung erzielt werden kann.



(a) (b)

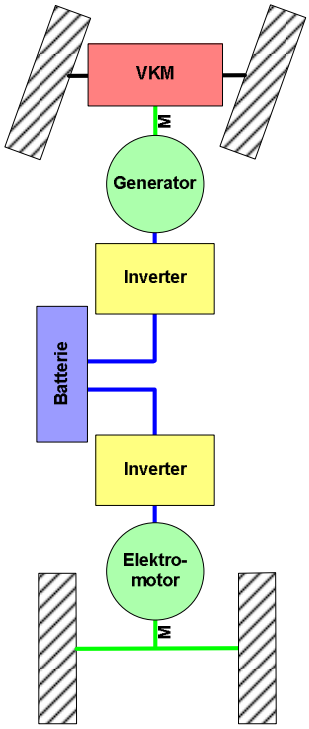
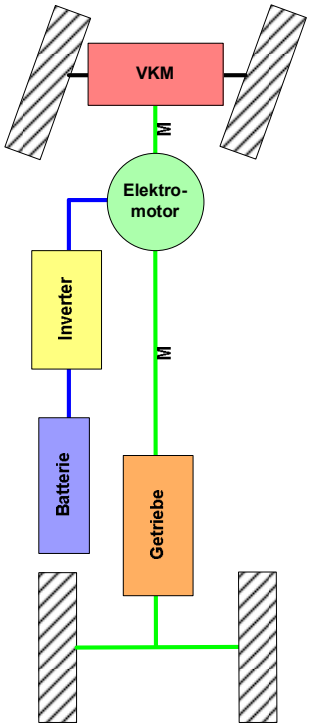
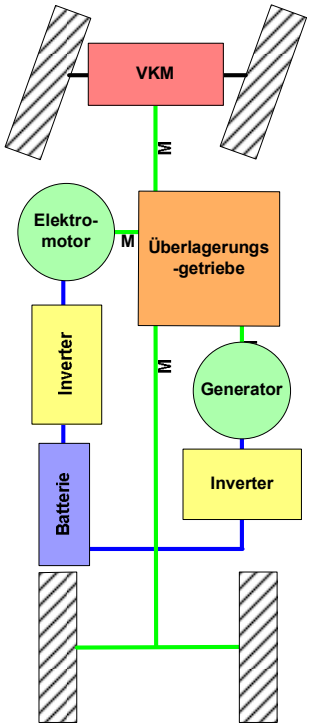
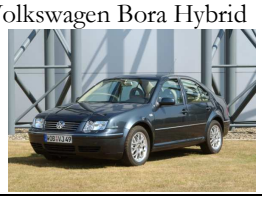
Abbildung 3.1: Prinzipdarstellung des Antriebstranges in Form eines Blockschaltbildes eines
(a) Elektroautos und eines
(b) Brennstoffzellenfahrzeuges.
Die Pfade der Momentenübertragung sind grün gezeichnet. Die Pfade der Übertragung der elektrischen Energie sind symbolhaft blau gezeichnet.

Hybridisierung von Antriebssträngen von Kraftfahrzeugen bedeutet, daß die Momentenerzeugung nicht mehr ausschließlich verbrennungsmotorisch erfolgt. Neben dem konventionellen verbrennungsmotorischen Antrieb kommt ein oder kommen mehrere alternative Antriebe zum Einsatz, die jeweils Beiträge zu dem Gesamtmoment auf der Kardanwelle bzw. Beiträge zur Gesamtenergiebilanz des Fahrzeuges leisten.

Hierbei wird unterschieden in:

- serielle Hybridantriebe, sogenannte Serienhybride,
- parallele Hybridantriebe, sogenannte Parallelhybride sowie
- leistungsverzweigte Antriebe, sogenannte Mischhybride

je nachdem, wie das zusätzliche Moment in den Antriebstrang eingeleitet wird (s. Tabelle 3.1).

	Serienhybrid	Parallelhybrid	Mischhybrid
Prinzipbild (Beispiele)			
Funktionsweise	• Generator wandelt mechanische Energie des VKM in elektrische Energie • Zwischenspeicherung der el. Energie in der Batterie • Generieren des Antriebsmomentes durch einen Elektromotor	• VKM und Elektromotor können alternativ oder gleichzeitig Momentenbeiträge zum Gesamtvortriebsmoment liefern	• es können beide Funktionsweisen, die des seriellen und des parallelen Hybridfahrzeuges realisiert werden (sogenannte **Leistungsverzweigung**)
Vorteile	• der VKM kann bezüglich seines Wirkungsgrades im Bestpunkt betrieben werden • erhöhter Komfort, da kein mehrstufiges Getriebe erforderlich	• höherer Wirkungsgrad als serieller Hybrid • geringste Kosten bei vergleichbarer Leistungsfähigkeit, da weniger und kleinere Bauteile	• volle Freiheit bei der Momentenerzeugung- und -verteilung durch die Betriebsstrategie (z. Bsp. radmomentbasiert oder wirkungsgradoptimal)
Nachteile	• schlechter Gesamtwirkungsgrad durch mehrfache Energiewandlung • erhöhte Kosten da doppelte Anzahl von Komponenten erforderlich (E-Motor, Inverter)	• möglicherweise Komfortnachteile durch das Schaltverhalten des Getriebes	• erhöhte Kosten da doppelte Anzahl von Komponenten erforderlich; Komponenten allerdings teilweise kleiner dimensionierbar als beim Serienhybrid
Beispiel	Volvo FL6 Hybrid 	Volkswagen Bora Hybrid 	Toyota Prius II 

Tab. 3.1: Klassifizierung der Hybridfahrzeuge. Die Pfade der Momentenübertragung sind grün gezeichnet. Die Pfade der Übertragung der elektrischen Energie von der Batterie zur Elektromaschine sind symbolhaft blau gezeichnet.

Weiterhin wird nach dem Leistungsverhältnis des zusätzlichen Antriebes zum konventionellem Antrieb unterschieden in Starter-Generator-Fahrzeuge und vollhybridische Fahrzeuge, sogenannte Vollhybride (s. Tabelle 3.2).

Die Ford Company, USA, hat in den 90-ern den Begriff des ‚Mild-Hybrid' bei hybridischen Kraftfahrzeugen geprägt. Diese weitere Unterteilung ist bezüglich der Zuordnung zu bestimmten Leistungsklassen unscharf und bis heute nicht klar definiert worden. Von der Arbeits- und Funktionsweise ist der ‚Mild-Hybrid' eher dem Starter-Generator-Fahrzeug zuzuordnen. Eine Unterscheidung der Hybridfahrzeuge hinsichtlich der Leistungen der im Fahrzeug verwendeten Elektromotoren ist eher schwer und nicht allgemein anerkannt. So tauchen in jüngster Vergangenheit immer weitere Unterscheidungen der Hybridfahrzeuge auf, wie etwa Microhybrid oder Halbhybrid, wobei nicht jede weitere Unterscheidung unbedingt sinnvoll ist [Bosch 2005, Maxwell 2005].

	Starter-Generator (SG) bzw. Mild-Hybrid (MH)	**Vollhybrid**
Arbeitsweise	VKM arbeitet immer	VKM und EMA arbeiten alternativ oder gleichzeitig
Funktionen	• VKM-Anlassen • Generatorfunktion • Start-/Stopp-Funktion • Rekuperation geringer Leistungen • Boosten	• VKM-Anlassen • Generatorfunktion • Start-/Stopp-Funktion • Rekuperation hoher Leistungen • Boosten • elektrisches (An-) Fahren
Potential zur Verbrauchseinsparung	< 9 % für SG; < 20 % für MH	> 20 %
Beispiel	Honda Insight 	Ford Escape 

Tab. 3.2: Vergleich von Starter-Generator-Fahrzeug und Vollhybrid
Grundsätzlich für diese Klassifizierung sind die Arbeitsweisen und das Vorhandsein der Funktion des elektrischen Fahrens.

Bei beiden Konzepten gibt es zwei mögliche Umsetzungen:

- Die Elektromaschine ist für alle Zeit mechanisch fest mit der Kurbelwelle verbunden. Diese Anordnung wird mit **kurbelwellenfest** bezeichnet. Man spart sich dabei eine Kupplung zwischen Verbrennungsmotor und Elektromaschine. Im Gegenzug jedoch kann aufgrund der Verbrennungsmotorschleppmomente weniger mechanische Energie rekuperiert werden, da diese beim freirollenden Fahrzeug immer vorhanden sind.

- Beim sogenannten **Impulsstarter** ist die Elektromaschine nicht für alle Zeit fest mit der Kurbelwelle verbunden. Es wird eine weitere Kupplung zwischen VKM und Elektromaschine benötigt. Im Gegenzug kann mehr Energie rekuperiert werden.

Der Antriebstrang, der in der vorliegenden Gesamtfahrzeugsimulation modelliert wurde, ist der Antriebstrang eines Volkswagen Bora Hybrid, welches in der Volkswagen Konzernforschung im Rahmen des Projektes SUVA aufgebaut wurde [Köhle 2004]. Aus diesem Grund war ein Zugriff auf die Daten und das Fahrzeug zwecks Vermessung und Validierung der Simulation gegeben.

Im Kapitel 3.1 werden der Aufbau und die Zielwerte (Verbrauch, Masse, Beschleunigung) des Volkswagen Bora Hybrid beschrieben. Im Kapitel 3.2 wird auf die Modellierung der Regelstrecke eingegangen. Die Modelle der Controller werden im Kapitel 3.3 kurz erläutert. In Kapiteln 3.4 wird auf die Gesamtintegration der Fahrzeugsimulation in Matlab/Simulink beschrieben.

3.1 Hybridfahrzeug Volkswagen Bora Hybrid im Projekt SUVA

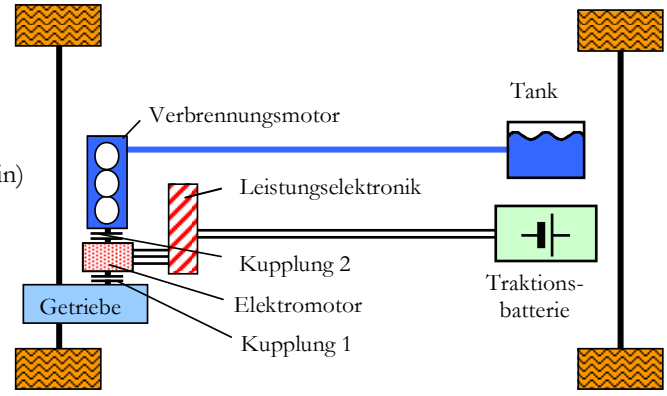
SUVA ist die Abkürzung für Surplus Value Hybrid – ein Fahrzeug mit einem hybridischen Antriebstrang, das neben seiner Verbrauchsreduzierung mit einem Mehrwert für den Käufer aufwarten kann, wie z. Bsp. besseren Fahrleistungen durch die elektrische Boostmöglichkeit oder besseren Komfort durch eine Standklimatisierung.

Das Projekt wurde durch die EU gefördert [Biermann et al. 2004] und dauerte von 2001 bis 2003. Die Abschlußpräsentation fand im Sommer 2004 in Prag gemeinsam mit den Projektpartnern Daimler-Chrysler AG und Fiat statt.

Der Antriebstrang des Volkswagen Bora Hybrid ist in der Abbildung 3.1.1 schematisch dargestellt. Bei diesem Parallelhybrid handelt es sich systembedingt um einen Vollhybrid.

<u>Parallelhybrid</u>

- 3-Zyl. TDI, 1,4 l 55 kW
- 2 Kupplungen
- Doppelkupplungsgetriebe DSG
- Elektromotor
  Spitzenleistung: 25 kW, 110 Nm (3 min)
  Dauerleistung: 8 kW
  Gewicht: 35 kg
- Batterie
  Nickelmetallhydrid NiMH
  Spitzenleistung: 30 kW
  Nennenergieinhalt ~2 kWh
  Gewicht: 60 kg

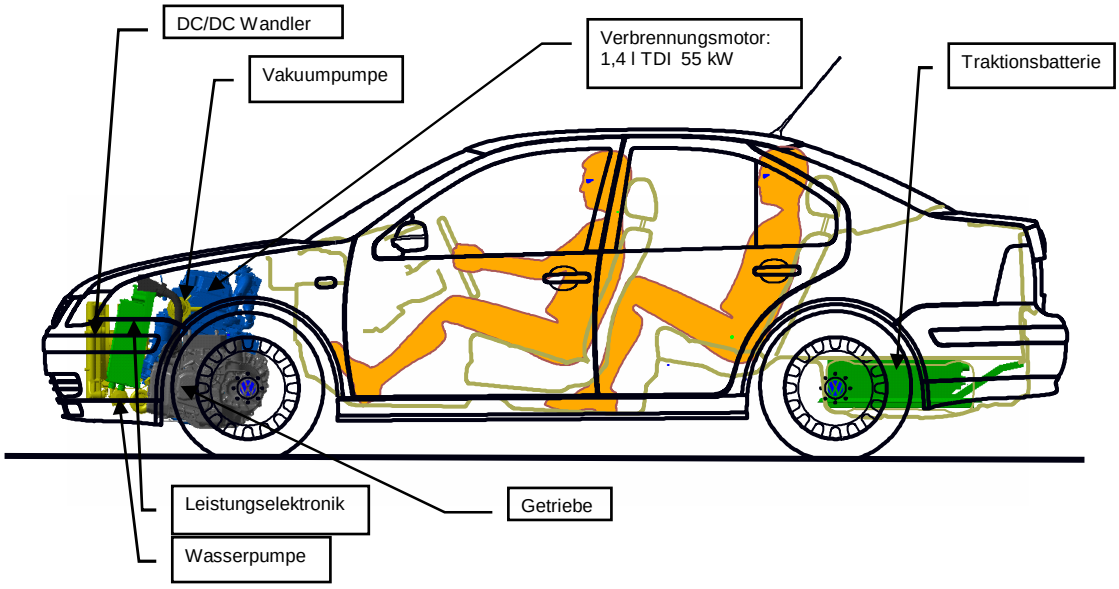


(Quelle: [Koehle 2004])

Abbildung 3.1.1: Kenngrößen und Prinzipdarstellung des Antriebstranges des Volkswagen Bora Hybrid im Projekt SUVA; Hierbei wurde die Kupplung 1, die tatsächlich die Doppelkupplung des DSG ist, zum besseren Verständnis der Betriebsmodi des Parallelhybrid außerhalb des Getriebes gezeichnet.

Aufgebaut ist der Volkswagen Bora Hybrid mit einer 55 kW TDI Verbrennungskraftmaschine mit Pumpe-Düse-Technik und einem 25 kW Elektromotor. Im Projekt wurden wahlweise eine Drehstrom-Asynchronmaschine und eine permanentmagneterregte Synchronmaschine als Elektromaschine verwendet. Das Batteriesystem besteht aus einer 288V, 6,5 Ah Nickel-Metallhydrid Batterie. Das verwendete Getriebe ist das innovative Doppelkupplungsgetriebe DSG©, ein Automatikgetriebe der Volkswagen AG (s. Kapitel 3.2.1).

Das Packaging des Volkswagen Bora Hybrid ist in der Abbildung 3.1.2 dargestellt.



(Quelle: [Koehle 2004])

Abbildung 3.1.2: Packaging des Volkswagen Bora Hybrid im Projekt SUVA. Die für die Simulation relevanten Komponenten sind farbig gekennzeichnet und beschriftet.

Das Verbrauchsziel des Projektes SUVA war < 4 Liter Diesel im NEFZ-Testzyklus beim warm gefahrenen Fahrzeug. Dabei sollten die Eckdaten des SUVA-Hybridfahrzeuges ungefähr denen des Vergleichsfahrzeuges entsprechen, d.h. der Fahrer sollte die eben genannte Verbrauchseinsparung bei gleicher Fahrleistung und gleichem Fahrkomfort erhalten (Tabelle 3.1.1).

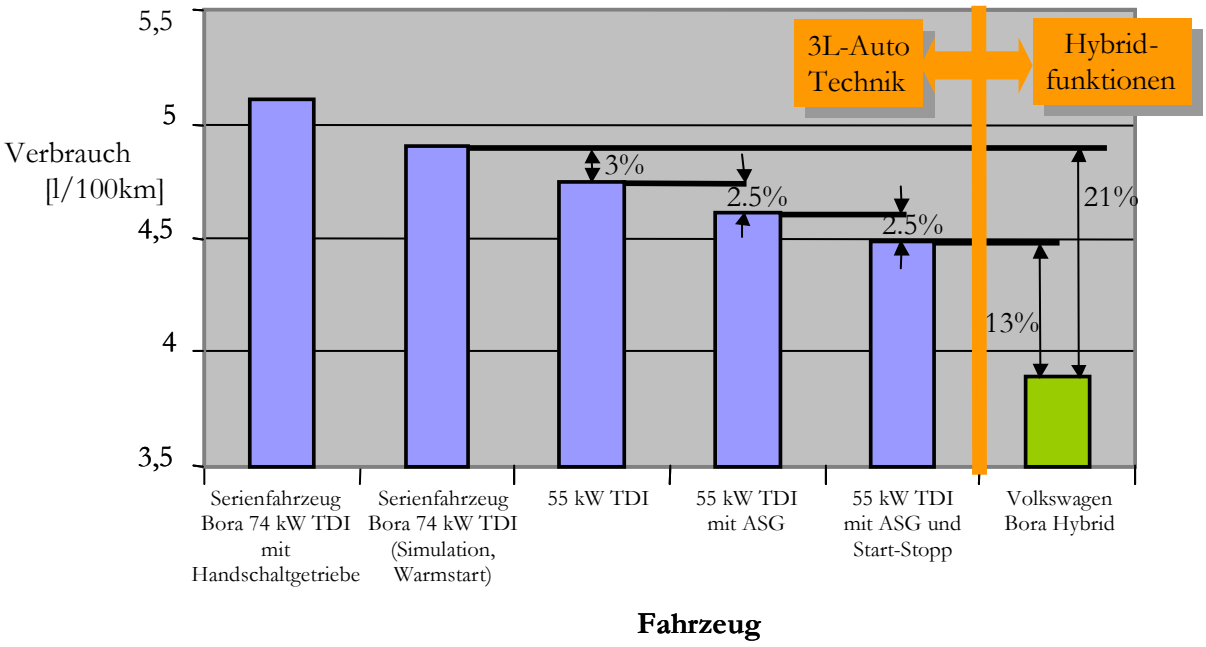
	VW Bora 1.9l TDI mit 5-Gang Handschaltgetriebe	VW Bora 1.9l TDI Automatik	VW Bora 1.4l TDI Hybrid
Leistung V- Motor [kW]	74	74	55
Leistung E- Motor [kW]	-	-	25
Beschleunigung von 0 auf 100 km/h [sec]	12.1	13.5	11
Höchstgeschwindigkeit [km/h]	188	184	172
Zuladung [kg] (je nach Ausstattung)	553 - 403	554-409	400
Kofferraumvolumen [l]	455	455	430
Verbrauch im NEFZ [l /100 km]	5,1	6,5	3,9
Klimaanlage	Ja	Ja	Ja
Servolenkung	Ja	Ja	Ja
Schaltung	5-Gang Handschaltung	5-Gang Automatik	6-Gang Automatik
Schadstoffeinstufung	Euro IV	Euro III	Euro IV

(Quelle: [Koehle 2004])

Tabelle 3.1.1: Zielwerte des Volkswagen Bora Hybrid im Vergleich: erste Spalte: Ausgangspunkt bzw. Vergleichsfahrzeug; zweite Spalte: Vergleichsfahrzeug mit Automatikgetriebe; dritte Spalte: Hybridfahrzeug SUVA mit leistungsschwächerem Verbrennungsmotor (sogenanntes Downsizing), Elektromotor und Automatikgetriebe

Die Verbrauchseinsparung größer 20 % wird erreicht durch folgende Funktionen (s. Abbildung 3.1.3):

- Downsizing der Verbrennungskraftmaschine von 74 kW auf 55 kW bzw. von 1,9 l TDI auf 1,4 l TDI,
- Ersetzen des Handschaltgetriebes durch ein Automatikgetriebe,
- Start-/Stoppbetrieb in den Standphasen,
- elektrisches Fahren und Lastpunktanhebung sowie
- Energierückgewinnung durch Schubrekuperation.



(Quelle: [Koehle 2004])

Abbildung 3.1.3: Einfluß der Maßnahmen zur Verbrauchsreduzierung beim Volkswagen Bora Hybrid im Vergleich (ASG: automatisiertes Schaltgetriebe)

3.2 Modell der Regelstrecke

Für die einzelnen Controller des Fahrzeuges ist die Regelstrecke das System Fahrzeug-Straße-Umwelt. Wie schon erwähnt, wurde das Modell der Regelstrecke in Modelica/Dymola modelliert. Simuliert wird die Regelstrecke mit variabler Schrittweite. Hierbei handelt es sich um eine Vorwärtssimulation (s. Kapitel 2.6, Punkt 5).

Die Regelstrecke wurde gemäß der in [Tiller et al. 2003b] vorgeschlagenen Vehicle Modelling Architecture (VMA) in drei Hauptobjekte strukturiert (Abbildung 3.2.1):

- das Modell des Antriebstranges (engl. Powertrain),
- das Modell des Chassis, der Fahrwiderstände, der Räder einschließlich Radbremse und des Kontaktes mit der Straße sowie
- das Modell der Nebenverbraucher (engl. Auxiliaries).

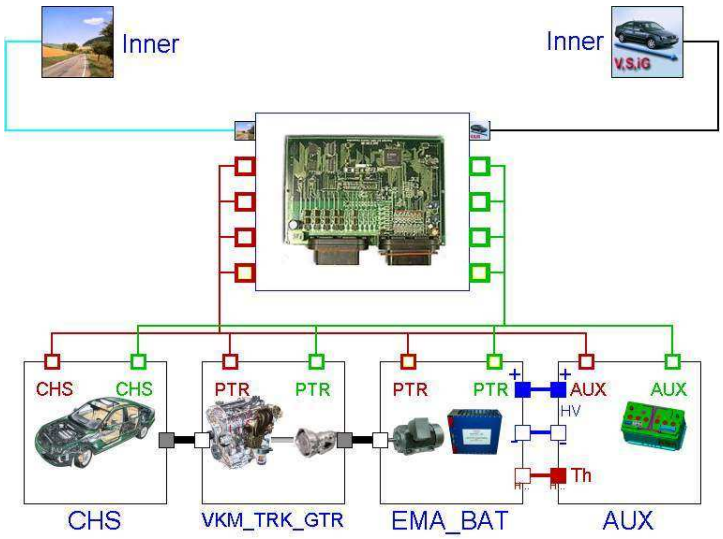


Abbildung 3.2.1: Oberste Modellebene des Dymolamodells der physikalischen Regelstrecke.
Die Objekte am oberen Rand stellen die Interfaces zur Umwelt dar (Temperatur, Steigung usw.). Die Objekte am unteren Rand enthalten die Modelle der physikalischen Regelstrecke und werden in den folgenden Kapiteln weiter detailliert. Das Objekt in der Mitte der Abbildung realisiert die Kopplung zum Mastermodell. Abkürzungen: CHS: Chassis, PTR: Powertrain, AUX: Auxiliaries (Nebenverbraucher wie Heizung, Klimaanlage, Licht, etc.), VKM: Verbrennungskraftmaschine, GTR: Getriebe, EMA: Elektromotorischer Antrieb, BAT: Batteriesystem der Traktionsbatterie (288 V Batteriesystem)

Das Modell des Antriebstranges enthält die Teilmodelle der momenterzeugenden Komponenten so zum Beispiel der Verbrennungskraftmaschine, der Elektromaschine, des Getriebes sowie des Traktionsbatteriesystems.

Im Modell der Nebenverbraucher werden die elektrischen Lasten des 12V-Bordnetz mit konstanten Leistungswerten modelliert sowie der DC/DC-Wandler, der die Batteriespannung auf Traktionspotential umsetzt auf 12 V für das Nachpuffern der Bordnetzbatterie.

Der Block in der Mitte der Abbildung 3.2.1 steht für die Anbindung des physikalischen Modells an das Mastermodell und beinhaltet das in Kapitel 4.3 beschriebene EXITE-Interface.

Ohne weiter an dieser Stelle auf die Details der Modellierung oder der Schnittstellen einzugehen sei noch erwähnt, daß das Teilmodell Chassis über die Achswelle mit dem Modell des Powertrains verbunden ist (schwarze Verbindung in Abbildung 3.2.1). Das Modell des Powertrains ist über die elektrischen Leitungen (blau) sowie über die thermischen Größen (braun) mit dem Modell der Nebenverbraucher verbunden. Die Regelstrecke wird mit variabler Schrittweite simuliert.

In den folgenden drei Kapiteln wird übersichtsartig auf die Modellbildung des Antriebstranges, des Systems Chassis und der Nebenaggregate eingegangen.

3.2.1 Modellierung des Antriebstranges

Das Modell des Antriebstranges wird aufgrund seiner Komplexität in die Aggregate gemäß der physikalischen Struktur des Antriebstranges strukturiert.

In den folgenden Unterpunkten wird die Modellierung der folgenden Aggregate kurz beschrieben:

- das Modell der Verbrennungsmotors,
- das Modell des Tanks,
- das Modell der Elektromaschine,
- das Modell der Trennkupplung,
- das Getriebemodell sowie
- das Modell des Traktionsbatteriesystems.

In einer frühen Version des dabei angewendeten Antriebstrangmodells wurden alle relevanten Aggregate in Dymola modelliert. Wie Laufzeituntersuchungen ergaben, war im Zusammenhang mit der Modelltiefe das Fahrzeugmodell nicht mehr handhabbar simulierbar (s. Kapitel 5.4). Die Simulation wurde extrem langsam ausgeführt. Die Modelle der Elektromotorsteuerung und des Elektromotors sollten aufgrund der hohen Abtastrate (s. Kapitel 3.3, Punkt 3) vorteilhalfterweise zusammen auf einem Rechner abgearbeitet werden. Aus diesem Grund wurden das Modell der Elektromaschine und der Traktionsbatterie aus dem physikalischen Modell des Fahrzeuges herausgelöst und zusammen mit der Elektromotorsteuerung in einem Matlab/Simulink-Modell modelliert.

1. Modell der Verbrennungsmotors

Das Modell des Verbrennungsmotors soll ausgehend von Momentenanforderungen und Kraftstoffzufuhr das mechanische Motormoment generieren. Die abgegebene Kraft der einzelnen Zylinder ist unter anderem abhängig vom Ladedruck und von den Strömungsverhältnissen im Auspuffrohr. Diese wiederum sind abhängig von dessen Gestaltung und Aufbau, also zum Beispiel vom Katalysator usw. Wenn man die Verbrennungskraftmaschine mathematisch modellieren möchte, dann müßte man von der Zündung ausgehend über die Bewegung der Kolben, die Druckausbreitung in den Abgasrohren bis hin zur Arbeitsweise des Katalysators den gesamten Abgasstrang modellieren. Um die Rechenbarkeit des Modells des VKM während eines vergleichsweisen langen Zyklus wie den NEDC zu gewährleisten sowie um den VKM auf einfache Weise für andere Fahrzeugkonfigurationen skalieren zu können, wurde das thermodynamische Modell des VKM durch ein Kennfeldmodell ersetzt, welches das Kraftstoffvolumen pro Arbeitstakt als Funktion des Motorsollmomentes und der Motordrehzahl berechnet. Modelliert wurde auf einfache Weise der Zusammenhang zwischen der Einspritzung und dem indizierten Moment, d.h. das theoretische Motormoment ohne Reibung. Durch Abziehen des Verlustmomentes vom indizierten Moment wurde das tatsächliche Antriebswellenmoment gebildet.

2. Modell des Tanks

Der Tank wurde als einfaches Durchflußmodell modelliert und ist hinreichend beschrieben durch die Volumenänderung pro Zeit. Somit ist der Durchfluß die einzige Schnittstellengröße zum Modell des Verbrennungsmotors.

3. Modell des elektromotorischen Antriebes

Das Modell des elektromotorischen Antriebes umfaßt die Komponentenmodelle Elektromaschine und Umrichter. Es gab drei Entwicklungsstufen bei der Modellbildung des elektromotorischen Antriebes, die an dieser Stelle von der Modellierung her kurz beschrieben werden sollen. Der Einfluß auf die Performance der Simulation wird in Kapitel 5.4 erläutert.

(EMA 1a) *Modell des elektromotorischen Antriebes in Dymola modelliert; Umrichter detailliert modelliert*
Zu Beginn dieser Arbeit wurde der elektromotorische Antrieb in Dymola modelliert: Die Elektromaschine, eine Asynchronmaschine, wurde dreiphasig beschrieben. Das dreiphasige System mit angeschlossenem Sternpunkt ist aufwendiger in der Beschreibung, stellt aber den allgemeinen Fall dar und kann auch unsymmetrische Maschinen abbilden.
[Leonhard 2000]
Der Umrichter besteht aus einem relativ exakten Modell des 6 kHz - Unterschwingungsverfahrens, eines elektrischen Modells einer B6-Brücke sowie eines dreiphasigen Stromsensors. Zwei PWM-Signale der Elektromotorsteuerung, die Sollstrangspannungen $U_{a,soll}$ und $U_{b,soll}$, werden dem Umrichter zugeführt und über das Unterschwingungsverfahren, der sogenannten Sinus-Dreieck-Modulation, in Ansteuersignale für die B6-Brücke gewandelt. Der Umrichter hat keinen Zwischenkreis mit einer konstanten Spannung, sondern ist eingangsseitig an die sich ändernde Spannung der Traktionsbatterie angeschlossen.

Diese Modellierung bildet zwar die Realität bezüglich des Umrichters sehr genau ab, war aber aufgrund der Dynamik des Umrichters nicht rechenbar.

(EMA 1b) *Modell des elektromotorischen Antriebes in Dymola modelliert; Umrichter abstrahiert*
Das Modell des Umrichters wurde dahingehend vereinfacht, daß nicht mehr die PWM-Signale der Elektromotorsteuerung dem Umrichter zugeführt werden, sondern zur PWM analoge Sollspannungssignale. Somit fallen die hochfrequente Übertragung der PWM-Signale aus dem Matlab/Simulink-Modell der Elektromotorsteuerung sowie die Nachbildung des Unterschwingungsverfahrens weg und es wird die Reaktion des Umrichters auf virtuelle PWM-Signale simuliert. Die Amplitude der Strangistspannungen ist durch die Spannung der Traktionsbatterie beschränkt. Die Verluste des Umrichters werden in der Form berücksichtigt, daß je Phase ein konstanter Widerstand angenommen wird, der die internen Verluste des Umrichters abbildet. Die Strombelastung der Traktionsbatterie wird aus der Leistungsbilanz (dreiphasige Spannung und dreiphasiger Strom) berechnet. Die Umschaltvorgänge der Leistungselektronik werden nicht berücksichtigt. Die Ansteuersignale der Elektromotorsteuerung in Matlab/Simulink werden über eine shared-memory-Kopplung zu dem Fahrzeugmodell übertragen, welches auf einem zweiten physikalischen Prozessor eines Doppelprozessorsystems abgearbeitet wird. Die Schaltfrequenz und somit die Frequenz der Übertragung beträgt wahlweise 8 kHz oder 4 kHz (s. Kapitel 5.4).

Diese Vereinfachung hat die Simulation rechenbar gemacht allerdings ca. 400 Mal langsamer als Echtzeit (Erläuterungen dazu s. Kapitel 5.4). Diese Abstraktion kann hinsichtlich der damit verbundenen Modellvereinfachungen hingenommen werden, wenn man nicht an den Auswirkungen der Schaltvorgänge auf die Momentenentwicklung im Antriebstrang interessiert ist. Der Fokus dieser Gesamtfahrzeugsimulation war nicht, eine Simulationsumgebung zu schaffen für die Entwicklung von Umrichtern oder Elektromotorsteuerungen. Wenn dies der Fokus ist, dann muß auch der Umrichter und das Schalten detailliert modelliert werden.

(EMA 2) *Modell des elektromotorischen Antriebes und der Traktionsbatterie in Simulink modelliert*
In den Modellen **(EMA 1a)** und **(EMA 1b)** wurde jeweils die Regelstrecke, also das gesamte Fahrzeug, in Dymola modelliert. Hintergrund war das Ziel der geschlossenen Lösung des Differentialgleichungssystems der Regelstrecke. Allerdings ist der Prozessor überfordert aufgrund der notwendigen, kleinen Simulationsschrittweite in der Sollstrangspannungsvorgabe durch die Elektronmotorsteuerung mit einer Frequenz von 8 kHz oder 4 kHz. (Zum Thema Solver und time-event s. Kapitel 4.7. und 5.4.) Beide Systeme, das Fahrzeugmodell und das Modell der Elektromotorsteuerung können aus Laufzeitgründen nicht auf einem Prozessor abgearbeitet werden, wenn die Performance der Simulation weiter erhöht werden soll. Somit wird in der Modellierung **(EMA 2)** auf die geschlossene Lösung der Differentialgleichungen des physikalischen Systems verzichtet und das ***elektrische Modell der Elektromaschine und der Traktionsbatterie*** aus dem Modell der Regelstrecke in Dymola herausgelöst. Beide Teilmodelle wurden in Matlab/Simulink umgesetzt und werden nun in einem Modell zusammen mit der Elektromotorregelung (Feldorientierte Regelung in d-q-Koordinaten) abgearbeitet. Da das umgebende elektrische Modell dreiphasig ist, müssen die Ströme vom α-β-Koordinatensystem umgerechnet werden in das dreiphasige a-b-c-Koordinatensystem der Wicklungsstränge. Da es keine nennenswerte Unterschiede bezüglich des Fahrzeugverhaltens bei den Schaltfrequenzen 8 kHz und 4 kHz gab, wurde dieses Matlab/Simulink-Modell mit 4 kHz fixed-step Integration gerechnet (ab Simulation (8) in Tabelle 5.4.1). Was die Simulation so viel schneller macht, ist der Wegfall des hochfrequenten Datenaustauschs zwischen dem Fahrzeugmodell in Dymola und dem Modell der Elektromotorsteuerung in Matlab/Simulink, da die nun realisierte shared-memory-Kopplung die Daten nur mit einer Zykluszeit von 10 ms austauscht. In Dymola wird über die Kurbelwelle die mechanische Reaktion des Elektromotors eingekoppelt. Die Outputsignale aus der Regelstrecke in Dymola als Inputs für das Simulinkmodell der Elektrik auf Traktionsseite sind die Schaltstellungen der drei Batterieschütze Hauptschütz 1, Hauptschütz 2 sowie des Vorladeschützes, Istdrehzahl der Verbrennungsmotorwelle, die Temperatur der Traktionsbatterie sowie die Spannung am DC/DC-Wandler, der das 12 V-Bordnetz stützt. Inputs der Regelstrecke in Dymola und somit Outputs aus dem Simulinkmodell der Elektrik auf Traktionsseite sind die Istladungsmenge und Leistung der Traktionsbatterie, der Ladestrom der von der Traktionsbatterie in den DC/DC-Wandler fließt, Traktionsbatteriestrom und -spannung sowie das mechanische Elektromotoristmoment. Die Modelle des DC/DC-Wandlers, der das Bordnetz aus der Traktionsbatterie puffert und das der Kühlung der Traktionsbatterie bleiben Bestandteil des Dymolamodells.

4. Modell der Trennkupplung

Die Modellierung der Zustände der Kupplung sowie der Übergänge zwischen den einzelnen Zuständen wurden der Beschreibung des Verhaltens der Kupplung nachempfunden, die durch den Kupplungshersteller geliefert wurde. Teil der Modellierung der Kupplung ist somit auch die elektrische Ansteuerung, die Kupplungsreibung sowie die hydraulische Aktuatorik. Die Trennkupplung ist fest mit dem Zweimassenschwungrad (ZMS) im Antriebstrang verbunden und bildet mit ihm eine Einheit. Aus diesem Grund wurde als weiteres Teilmodell innerhalb des Trennkupplungsmodells das Zweimassenschwungrad modelliert. Das ZMS besteht aus einer eingangseitigen und einer ausgangsseitigen trägen Masse, die axial zueinander gelagert sind und die sich gegeneinander verdrehen können. Die Verdrehung wird über ein System von gebogenen Federn mit Anschlag begrenzt. Die Aufgabe des ZMS ist es, Antriebstrangschwingungen zu dämpfen. Es wurden einzelne Federbögen mit schub- bzw. zugabhängiger Reibung und einem Anschlag modelliert.

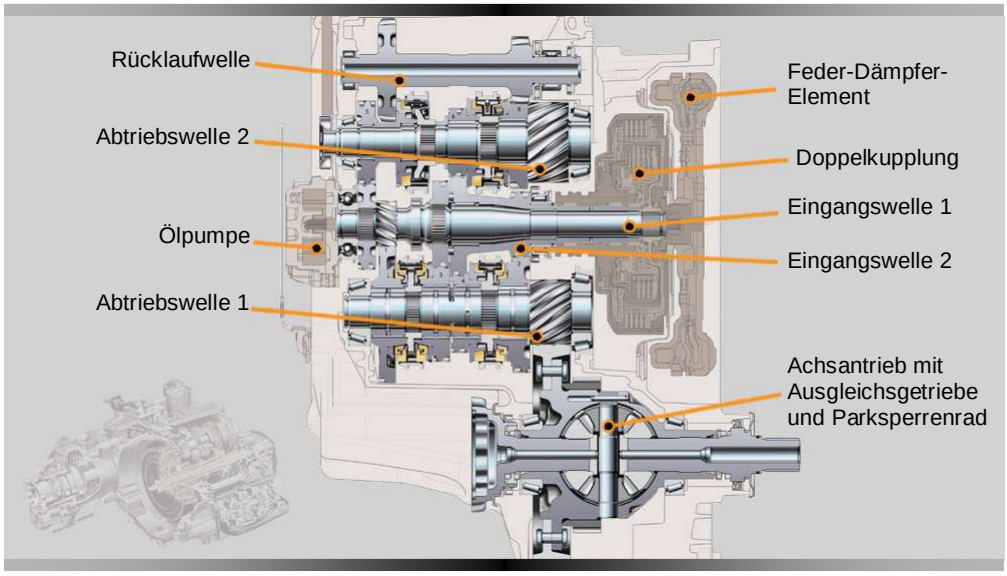
5. Modell des Traktionsbatteriesystems

Das elektrothermische Modell des Batteriesystems umfaßt die Modellierung des Systems Traktionsbatterie einschließlich aller Schütze, Sensorik und der Kühlung.

6. Getriebemodell

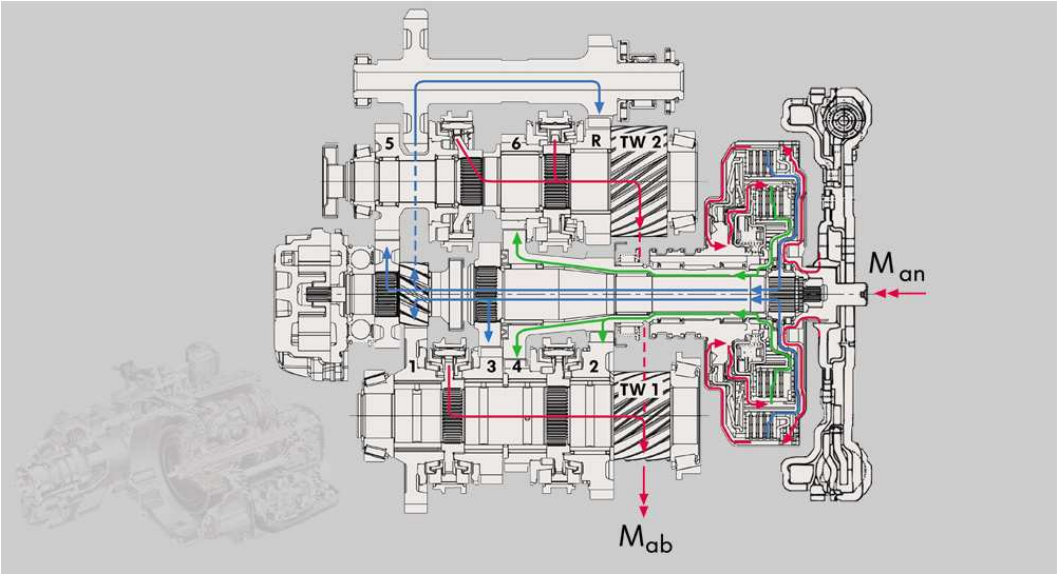
Das Direktschaltgetriebe der Volkswagen AG ist ein relativ neues, außergewöhnliches Automatikgetriebe, welches seit 2003 gefertigt wird. Funktionsweise ist beschrieben in [Pape 2004]. Da in Kapitel 5 an verschiedenen Stellen auf die Problematik der Auswirkung der Modellierungstiefe und -güte auf die Performance der vorliegenden Gesamtfahrzeugsimulation eingegangen wird, soll hier das Modell des DSG etwas detaillierter beschrieben werden.

(GTR 1) *Getriebemodell als Modell des DSG*
Das DSG besitzt im Gegensatz zu konventionellen Schaltautomaten zwei Eingangswellen (s. Abbildung 3.2.2 und Abbildung 3.2.3). Ist ein Gang eingelegt und die entsprechende Kupplung geschlossen, wird das Moment der Eingangswelle übertragen, währenddessen auf der anderen Welle die nächste Gangstufe bereits voreingestellt wird. Das Schalten des Momentes von einer Eingangswelle auf die andere erfolgt über die Kupplungsregelung der Doppelkupplung Kupplung 1/Kupplung 2, die sogenannte Kupplungsüberschneidung, weshalb dieses System auch als zugkraftunterbrechungsfrei schaltend bezeichnet wird.



(Quelle: [Pape 2004])

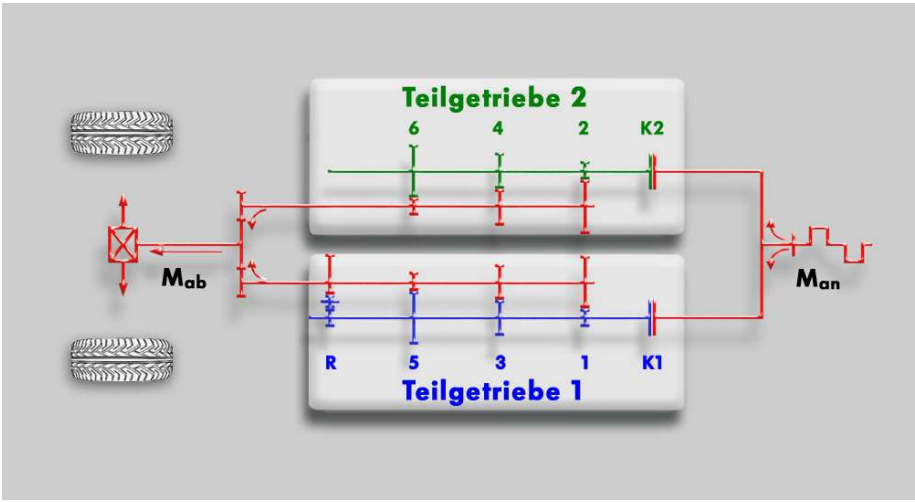
Abbildung 3.2.2: Schnittbild des DSG



(Quelle: [Pape 2004])

Abbildung 3.2.3: Schnittbild des DSG; eingezeichnet sind die Kraftflüsse der Eingangswellen 1 und 2

Die Modellierung (s. Abbildung 3.2.5) entspricht dem theoretischen Ersatzschaltbild in Abbildung 3.2.4 mit den entsprechenden Synchronisierungseinrichtungen (hier mit Synchronkupplungen bezeichnet), der Gangstellerlogik (GS), den Wellen einschließlich der Trägheiten und Reibungen, der Hydraulik (HYD), den Sensoren und natürlich den jeweiligen Zahnrädern mit deren Übersetzungen.



(Quelle: [Pape 2004])

Abbildung 3.2.4: Das vereinfachtes Ersatzschaltbild des DSG in einem konventionellen Antriebstrang K1, K2 : Kupplung 1 bzw. 2 der Doppelkupplung; M_{an} : Antriebsmoment seitens der VKM-Welle; M_{ab} : Abtriebsmoment der Kardanwelle

Die Abbildung 3.2.5 zeigt deutlich, wie bei der Modellierung in Dymola die physikalische Struktur des Systems erhalten bleibt und gut sichtbar ist.

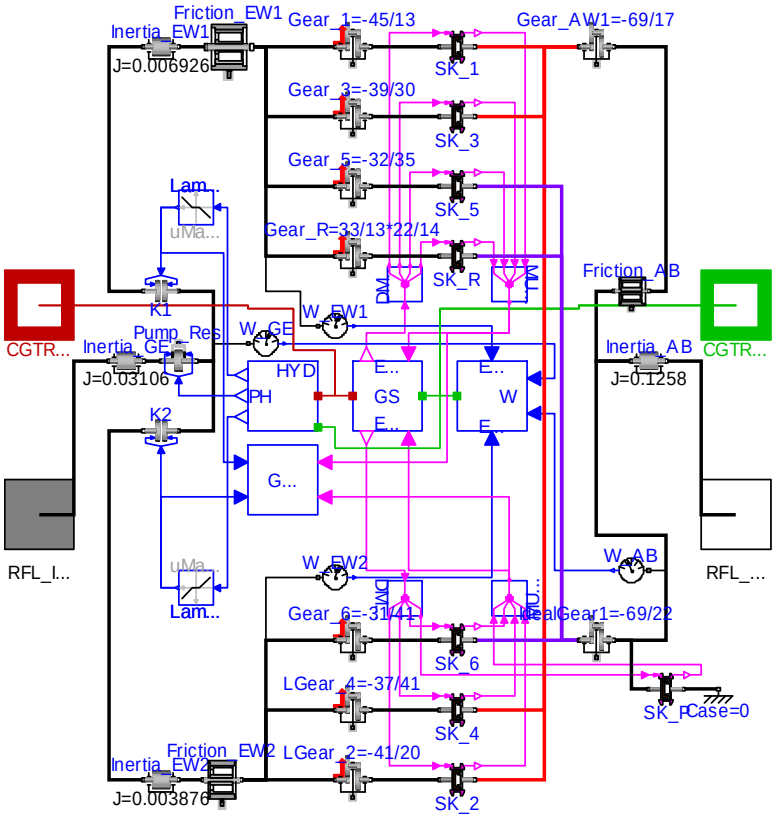


Abbildung 3.2.5: Dymolamodell des DSG, das gemäß den Konstruktionsunterlagen modelliert wurde

Die Verluste der einzelnen Zahnradsätze wurden drehzahlabhängig modelliert. Die für den Getriebecontroller relevante Sensorik und Aktuatorik wurden ebenfalls modelliert. Das Getriebemodell besteht aus einem elektro-hydraulischen Modell der Doppelkupplung und aus Modellen der Synchronisationseinrichtungen und der Gangsteller. Das thermische Verhalten des Automatikgetriebes wurde nicht modelliert.

Dieses Modell in Verbindung mit dem Getriebecontroller hatte in dieser Modellierungstiefe eher Nachteile bezüglich der Rechenperformance (s. Kapitel 5.4). Aus diesem Grund wurde ein idealisiertes Getriebemodell **(GTR 2)** modelliert, welches grundlegend das zugkraftunterbrechungsfreie Schalten abbildet.

(GTR 2) *Idealisiertes Schaltgetriebe*
Um das Phänomen der Kupplungsüberschneidung bzw. des zugkraftunterbrechungsfreien Schaltens abzubilden ohne bezüglich der serienmäßigen Aktuatorik reagieren zu müssen, wurde vom Prinzip her das DSG in ein Getriebe mit einer schaltbaren Übersetzung überführt (Abbildung 3.2.6). Damit sich das übertragene Moment beim Schalten nicht sprungartig ändert, wurde eine Kupplung eingeführt, die ein kontinuierlich veränderbares Moment übertragen kann.
Im unteren Zweig des Modells ist das Teilmodell des idealisierten Schaltgetriebes für die Rückwärtsfahrt des Fahrzeuges zu sehen. Die Sensorik des Getriebecontrollers wie im Modell **(GTR 1)** wird weiterhin bedient. Mit diesem Modell war es möglich, ohne negative Rückwirkungen des Getriebecontrollers das Fahrzeug virtuell fahren zu lassen.

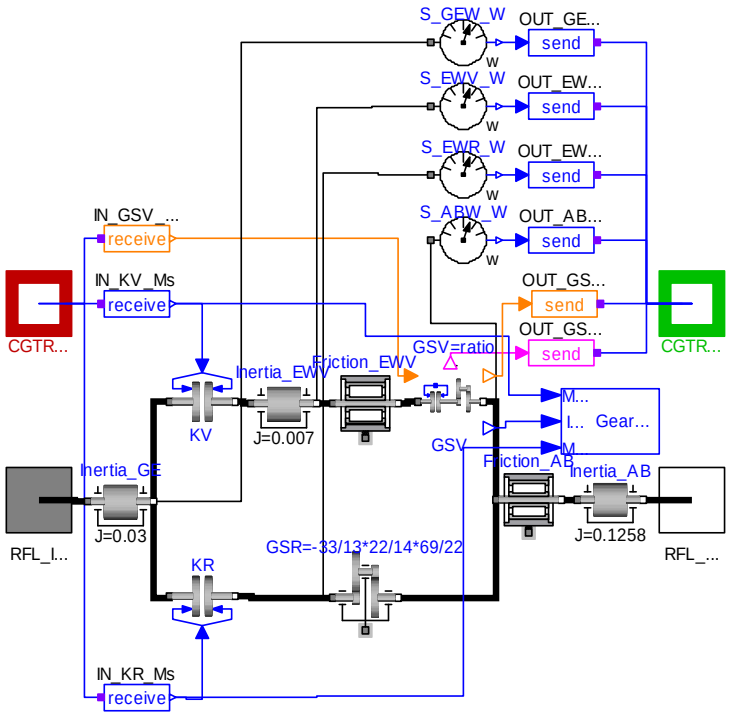


Abbildung 3.2.6: Dymolamodell des idealisierten Schaltgetriebes; Der obere Momentenpfad entspricht dem Vorwärtsgang und der untere Momentenpfad dem Rückwärtsgang

3.2.2 Modellierung des Systems Chassis

Das Modell des Chassis beinhaltet das Einspurmodell eines Fahrzeuges nach [Pruckner et al. 2001, Kramer 2002], ein Modell für den Kontakt Reifen-Straßen sowie ein einfaches Hydraulikmodell für die Bremse.

1. Einspurmodell des Fahrzeugs

Ein Einspurmodell ist für die Verbrauchs- und Reichweitensimulation von Hybridfahrzeugen ohne dem sogenannten Torque-vectoring, d.h. ohne Beeinflussung der Achsmomentverteilung, völlig ausreichend, wenn es um die Spezifikation des Energiemanagementcontrollers geht, da die Querdynamik hierbei eine untergeordnete Rolle spielt. Die Fahrwiderstände, die durch die Steigung der Straße und aufgrund der Frontfläche des Fahrzeugs charakterisiert sind, werden berücksichtigt. Die Problematik der Kraftübertragung bzw. des Kontaktes zwischen Reifen und Straße wurde in der Literatur in vielfacher Weise beschrieben [Reimpell et al. 1986, Mühlmeier 1993, Mitschke 1990]. Bekannt sind die vermessenen Seitenkraft-Traktionskennfelder (Überlagerung von Seiten- und Umfangskräften) – die sogenannten Krempel-Diagramme. Sie beschreiben die Seiten- bzw. Längskraft in Abhängigkeit vom Umfangsschlupf und Schräglaufwinkel. In der vorliegenden Gesamtfahrzeugsimulation wird nur die Längsdynamik simuliert und somit ist der Schräglaufwinkel null. Damit fällt eine Dimension, nämlich die der Reifenquerkräfte weg. Da dieses Diagramm individuell vermessen wird, trifft es nur für genau einen bestimmten Reifen zu. Überhaupt ist diese Problematik sehr vom verwendeten Reifen abhängig.

2. Reifenmodell

Um die Modellierung zu vereinfachen, wurde als Reifenmodell das Pacejkamodell gewählt [Pacejka 2002, Gaal 2003, Gipser 2000]. Die vergleichsweise einfache, analytische Funktion, auch „magic formula“ genannt, approximiert die realen Verhältnisse des Kontaktes Reifen-Fahrbahn hinreichend. Diese Formel ist Standard in der Fahrzeugsimulation und für viele Betrachtungen ausreichend. Teilweise werden allerdings weit mehr als 15 Koeffizienten verwendet. Der Nachteil dieser Approximation besteht in der Beschaffung der Koeffizienten bzw. darin, daß viele der Koeffizienten nicht physikalisch interpretiert werden können. In der vorliegenden Gesamtfahrzeugsimulation wurden nur die zwei Koeffizienten c und b verwendet (s. (Gl. 3.1)). Die Formel kann mit teilweise anderen Koeffizienten sowohl in longitudinaler als auch in lateraler Richtung angewendet werden. Da in der vorliegenden Simulation aus den oben genannten Gründen die Querdynamik nicht betrachtet wird, wurde hier nur die eindimensionale Form in longitudinaler Richtung für die Berechnung der Umfangskräfte benutzt.

Die „magic formula“ wurde reduziert auf folgende bekannte Formel:

$$F_x = F_N \cdot \mu \cdot \sin(c \cdot \arctan(b \cdot \lambda)) \qquad \text{(Gl. 3.1)}$$

mit
- F_x : Radkraft in longitudinaler Richtung
- F_N : Normalkraft am Rad
- μ : Reibbeiwert der Straße
- c , b : Schlupfkurvenparameter
- λ : Radschlupf

3. Hydraulikmodell

Das Hydraulikmodell der Bremse setzt die Bremsenanforderung des Bremsencontrollers über ein einfaches Hydraulikmodell in die mechanische Betätigung der Bremskolben um.

3.2.3 Modell der Nebenaggregate

Fahrzeugkomponenten, die zusätzliche Verbraucher im Sinne der Verbauchssimulation sind, werden in dem Modell der Nebenaggregate zusammengefaßt. Teilmodelle im System Nebenaggregate sind das Modell des Systems Heizung, Klima, Lüftung (HKL), das Modell des DC/DC-Wandlers, der sich elektrisch gesehen zwischen Traktionsbatterie und Bordnetzbatterie befindet und die Bordnetzbatterie mit einer zeitlich konstanten Grundlast (Glühlampen, Steuergeräte etc.).

Das System HKL besteht aus der Modellierung der verschiedenen Verbraucher wie elektrischer Heizung, Klimaanlage mit Kühlmittelpumpen und des Klimakompressors, die als konstante Lasten je nach Anforderung zugeschaltet werden können.

Der DC/DC-Wandler ist als steuerbarer, bidirektionaler Hoch- und Tiefsetzsteller gemäß seiner Spezifikation modelliert.

Die Bordnetzbatterie wurde über den Zusammenhang

$$U_{Bat} = N_{cell} \cdot (U_{cell} - R_{cell} \cdot I_{cell}) \qquad \text{(Gl. 3.2)}$$

modelliert.

Dabei bedeuten:

U_{Bat}	: Batteriespannung der Bordnetzbatterie
N_{cell}	: Anzahl der Zellen der Batterie
U_{cell}	: Zellspannung als Funktion des SOC
R_{cell}	: Zellwiderstand als Funktion des SOC
I_{Bat}	: Batteriestrom

Die thermischen Eigenschaften der Bordnetzbatterie wurden nicht modelliert.

3.3 Modelle der Steuergerätealgorithmen

Elektronische, mikrocontrollerbestückte Steuergeräte, ECUs, verarbeiten Sensorgrößen von anzusteuernden Aggregaten und geben über digitale oder analoge Ausgänge Steuersignale zur Beeinflussung dieser Aggregate an geeignete Aktuatoren weiter. Diese Steuergeräte werden auch Controller genannt. Jedoch auch die in den Steuergeräten ablaufenden Algorithmen, der Funktionscode, der in den meisten Fällen eine digitale Abtastregelung ist, wird Controlalgorithmus oder Controller (engl. für Regelung) genannt. Im Folgenden ist Controller immer auch gleichbedeutend mit Controlalgorithmus des jeweiligen Steuergerätes.
Sämtliche Controller in Matlab/Simulink werden diskret simuliert. Die Simulationsschrittweite ist fest und außer bei dem Elektromaschinencontroller und bei dem Batteriecontroller 10 ms. D.h. ereignisgesteuerte Algorithmen in den Controllern wurden nicht abgebildet und sind so auch nicht möglich abzubilden, da die hier vorgestellte Simulatorkopplung von einer festen Abtastrate in der Kommunikation ausgeht. Wollte man ereignisgesteuert Controller bedienen, müßte man mit der hier vorgestellten Methodik eine sehr kleine Abtastrate für die Kommunikation wählen, um dann synchron zu den Ereignissen die Algorithmen der Controller abzuarbeiten.
Die Algorithmen der Controller, die in dieser Gesamtfahrzeugsimulation simuliert werden sollen, enthalten bis auf das BMS in ihrer Funktionssoftware ausschließlich Größen im Datenformat double bzw. binäre Variablen. Somit fallen Normierungen weg. Dieser Weg wird auch in Zukunft zunehmend unterstützt insbesondere durch Softwarearchitekturen, wie sie in Autosar definiert und vorgeschlagen werden, um die reine Funktionssoftware klar zu trennen von der hardwarenahen ECU-Basissoftware. Im Teilmodell BMS wird wie auch im realen BMS-Steuergerät in Integer gerechnet. Die physikalischen Größen werden eingangsseitig umnormiert.

In den folgenden Unterpunkten werden die grundlegenden Funktionen der modellierten Controller kurz beschrieben ohne zu stark auf Details einzugehen.
Modelliert wurden:

- das Motorsteuergerät VKM-ECU,
- das Getriebesteuergerät GTR-ECU,
- die Elektromotorsteuerung EMA-ECU,
- der Energiemanagementcontroller VMU,
- der virtuelle Controller der Mensch-Maschine-Kommunikation HMI-ECU,
- der Controller des DC/DC-Wandlers DCW-ECU,
- das Bremsensteuergerät BRE-ECU sowie
- das Batteriemanagementsystem BMS.

1. VKM-Controller VKM-ECU

Der VKM-Controller, wie auch alle anderen im Folgenden in Simulink modellierten Steuergerätealgorithmen, ist gemäß der Abbildung 3.3.1 aufgebaut.

Im allgemeinen Fall kann der Controller in verschiedenen Simulationen benutzt werden und ist somit durch einen Link mit dem Modell einer Library verbunden. Es werden Anpassungsblöcke benötigt, um die Ein- bzw. Ausgangssignale des Controllers an die jeweilige Simulationsumgebung anzupassen. Diese Funktion übernehmen die Subsysteme

VKM_InputVSP bzw. VKM_OutputVSP in der Abbildung 3.3.1. Der Block VKM_Exite (s. Kapitel 4.3) stellt das Interface der VKM-ECU zum Mastermodell in der Clustersimulation dar. Hier werden die verwendeten Inputgrößen abgegriffen bzw. die Outputgrößen herausgeschrieben.

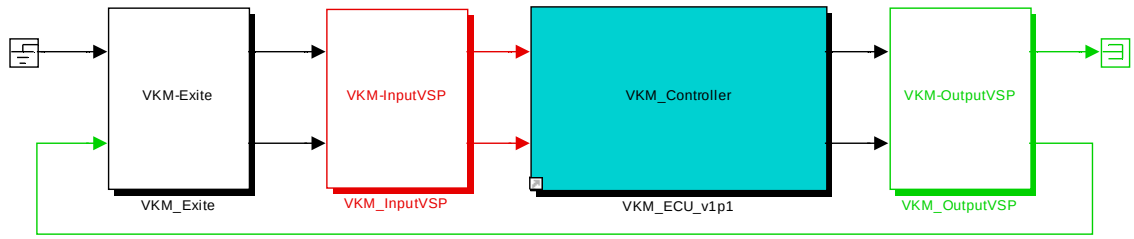


Abbildung 3.3.1: Allgemeiner Aufbau der in Simulink modellierten Steuergerätemodelle für die Clustersimulation. Eingebunden über einen Link auf das Modell VKM_ECU_1p1.mdl ist hier eine archivierte Version des VKM-Controllers.

Grundsätzlich sind alle in Simulink modellierten Controller aus folgenden Hauptmodulen aufgebaut (Abbildung 3.3.2):

- Basissicherheitssoftware BSS (Sicherheitsalgorithmen auf Basis noch unverarbeiteter Sensorgrößen)
- Signalvorverarbeitung SVV (Filterung, Differentiation usw. von Sensorgrößen)
- Sicherheitssoftware SIS (Sicherheitsalgorithmen auf Ebene verknüpfter, verarbeiteter Signale)
- Funktionssoftware FSW (eigentlicher Algorithmus, der für dieses Steuergerät typisch ist)
- Aggregatebedienung AGG (Algorithmen, die nicht typisch für dieses Steuergerät sind und ggf. andere Aggregate bedienen oder sogenannte Gatewayfunktionen übernehmen)

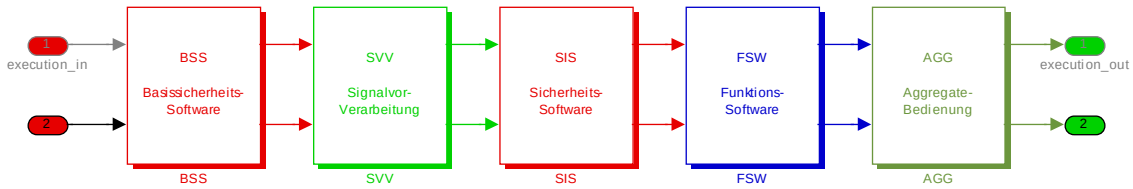


Abbildung 3.3.2: Allgemeiner Aufbau der in Simulink modellierten Controller mit seinen Hauptmodulen auf oberster Ebene.

In der Funktionssoftware des VKM-Controllers wird ausgehend von der Differenz zwischen der Solldrehzahl und der Istdrehzahl des Verbrennungsmotors durch den Drehzahlregler eine Solldrehbeschleunigung des Verbrennungsmotors berechnet. Aus dieser und einem inversen Modell der Mechanik des Verbrennungsmotors wird ein Sollmoment berechnet aus welchem dann die Einspritzmenge Kraftstoff abgeleitet wird, die dem Tank entnommen wird.

2. Getriebecontroller GTR-ECU

Der in Matlab/Simulink modellierte Controller des Getriebesteuergerätes wurde der DSG-Softwarespezifikation nachempfunden und wird in dieser Schrift aus Gründen der Geheimhaltung nicht näher erläutert. Er enthält unter anderem die Teilmodelle Fahrertyperkennung, Momentenberechnung, Gangbestimmung und Gangstelleransteuerung.

3. Controller der Elektromotorsteuerung EMA-ECU

Der Controller der Elektromaschinensteuerung beinhaltet im Wesentlichen die bekannte Vektorregelung der Asynchronmaschine [Leonhard 2000]. Die Schrittweite des fixed-step Solvers in der Simulation beträgt 25 µs.

4. Energiemanagementcontroller VMU

Der Energiemanagementcontroller beinhaltet neben einer Vielzahl von Ansteuerfunktionen für die Komponenten des Fahrzeuges den sogenannten Hybridmanager, der das eigentliche Verteilen der Momente auf VKM und Elektroantrieb koordiniert.

Der Hybridmanager realisiert die folgenden Funktionen:

- die Umsetzung der Geschwindigkeitsregelanlage , d.h. des sogenannten Tempomats,
- die Umsetzung des Fahrerwunsches in ein Getriebeeingangsmoment unter Berücksichtigung des Fahrzustandes,
- die Hybridstrategie, d.h. die VKM-An-/-Aus-Logik und die Momentenaufteilung zwischen dem Verbrennungsmotor und dem elektromotorischen Antrieb,
- die Momenten- und Gangvorgabe für Sonderfunktionen, wie Start, elektrisches Anfahren usw. sowie
- die Gangvorgabe für das Automatikgetriebe.

5. Mensch-Maschine-Interface HMI-ECU

Neben der Ansteuerung der momentenerzeugenden und -verteilenden Antriebstrangkomponenten wie zum Beispiel des Verbrennungsmotors oder des Getriebes durch die jeweiligen Controller gibt es Komponenten im Fahrzeug, die die Mensch-Maschine-Kopplung sicherstellen wie:

- das Kombiinstrument (Geschwindigkeitsanzeige, Öldruck usw.),
- der Gangwählhebel,
- der Betriebsartenschalter (Umschaltung zwischen reinem VKM-Betrieb, Hybrid-Betrieb oder reiner Elektrofahrt),
- das Zündschloss,
- die Geschwindigkeitsregelanlage sowie
- das Gas- und das Bremspedal.

Diese eben genannten Komponenten liefern Signale, die in der Antriebstrangsimulation für die entsprechenden Aggregate und Controller von Bedeutung sind. Aus diesem Grund werden die Grundfunktionen dieser Komponenten in der vorliegenden Gesamtfahrzeugsimulation modelliert und in einem virtuellen Controller HMI-ECU zusammengefaßt. Im

realen Fahrzeug sind diese Funktionen jedoch dezentral verteilt und teilweise in verschiedenen Steuergeräten oder aber in Hardware realisiert.

6. DC/DC-Wandler DCW-ECU

Im Volkswagen Bora Hybrid gibt es nicht mehr die keilriemengetriebene Fahrzeuglichtmaschine. Das Starten des Verbrennungsmotors und das Generieren der elektrischen Leistung erfolgt ausschließlich durch die Elektromaschine, die zwischen Verbrennungsmotor und Getriebe angeordnet ist (s. Abbildung 3.1.1). Die elektrische Leistung wird generiert durch Anhebung des abgegebenen Momentes des Verbrennungsmotors und Betreiben der Elektromaschine als Generator sowie durch Rekuperation in Freiroll- oder Bremsphasen. In beiden Fällen wird jedoch nur die Traktionsbatterie nachgeladen, so daß ein DC/DC-Wandler notwendig ist, um das Bordnetz zu stützen bzw. die Bordnetzbatterie nachladen zu können. Bei dem im Fahrzeug verbauten DC/DC-Wandler handelt es sich um einen regelbaren, mikrocontrollergesteuerten, bidirektionalen 2 kW DC/DC-Wandler, dessen Funktionsweise durch ein Lastenheft definiert ist. Das Modell der DCW-ECU entspricht der Spezifikation des Lastenheftes.

7. Bremsensteuergerät BRE-ECU

Das Modell des Bremsensteuergerätes BRE-ECU basiert im Wesentlichen auf einer einfachen Umsetzung der Funktionen ASR (Antriebsschlupfregelung im Fall der Beschleunigung) und EBV (elektronische Bremskraftverteilung im Bremsfall) des bekannten ABS-Systems. Die jeweiligen ASR- oder EBV-Eingriffe sollen die Frage klären, ob die simulierten Radkräfte überhaupt zum Vortrieb des Fahrzeuges beitragen oder ob nicht vielmehr oberhalb der Kraftschlußgrenze simuliert wird und die Räder ohne nennenswerten Vortrieb durchdrehen.

8. Batteriecontroller BMS

Das BMS ist in der hier vorliegenden Gesamtfahrzeugsimulation der einzige Controller, der nicht in Simulink modelliert worden ist, sondern in der Programmiersprache C programmiert worden ist und somit als DLL vorliegt. Die Architektur der BMS-Software ist modular so gestaltet, daß die Anwendersoftware, d.h. die Funktionssoftware klar getrennt ist von der Basissoftware (Abbildung 3.3.3). Diese Strukturierung wurde im Jahre 2000 begonnen, als es noch keine verwertbaren Empfehlungen von Seiten des Autosar-Konsortiums hinsichtlich der Strukturierung gab. Es ist darauf geachtet worden, daß die Anwendersoftware ohne funktionelle Einschränkungen separat auf jedem beliebigen Zielsystem abgearbeitet werden kann.

Im Entwicklungsprozeß der BMS-Software gibt es die Möglichkeit, mit verschiedenen Projektkonfigurationen ein Projekt zu kompilieren. Dabei bedeutet Projektkonfiguration das Anwenden eines bestimmten Compilers und eines bestimmten make-Files auf einen bestimmten Satz von Files. Es werden nicht immer alle Files in die Kompilation eingebunden. Bei der Projektkonfigurationen SIL wird eine DLL generiert für das Einbinden der BMS-Software in die hier vorliegende Simulation. Hierbei wird der Hardware-Abstraction-Layer des Target-Steuergerätes durch einen geeigneten Layer für die PC-Simulation ersetzt. Dadurch, daß mit der high-level-Basissoftware auch der Taskscheduler mit in die SIL hinein kompiliert wird, wird die BMS-Funktionssoftware mit dem in der Realität kooperativen Tasksystem (10 ms-, 100 ms-, 1 s-, 1 min – Task) identisch in der vorliegenden Antriebsstrangsimulation abgearbeitet.

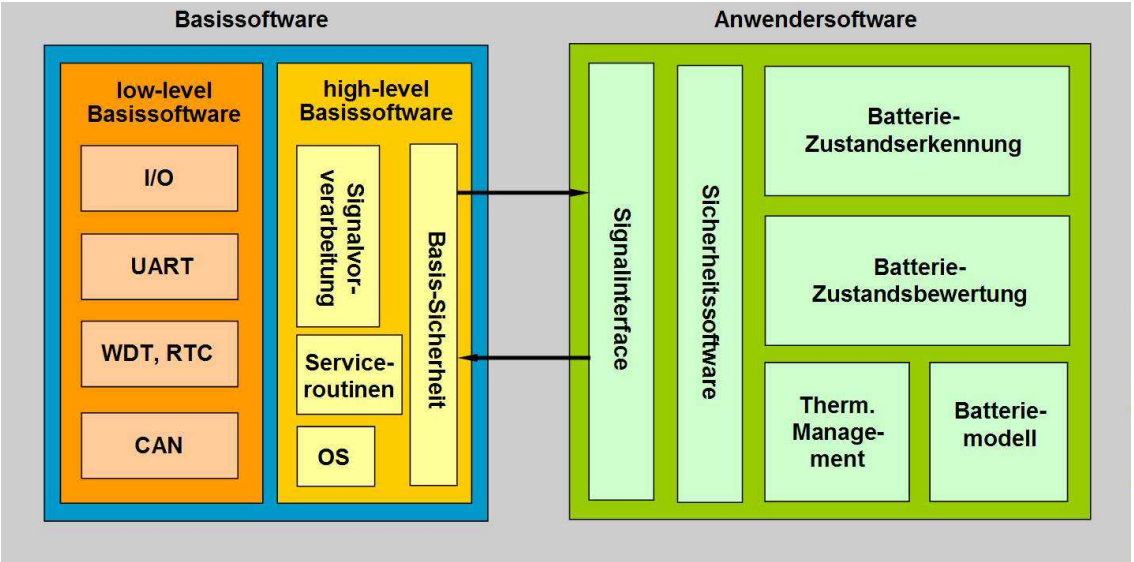


Abbildung 3.3.3: Prinzipdarstellung der Strukturierung der BMS-Software. Erkennbar ist die Trennung zwischen Basissoftware und Anwendersoftware sowie innerhalb der Basissoftware die Trennung in high-level und low-level Basissoftware

3.4 Gesamtintegration

Wie weiter oben schon des öfteren erwähnt, erfolgt die Gesamtintegration aller Modelle in Matlab/Simulink. Nachfolgend wird das Modell des Masters beschrieben, worin die Kopplung der einzelnen Teilmodelle untereinander erfolgt. Weiterhin werden das Modell des Monitors zur Aufzeichnung der Daten und das Modell DRV der Zyklusvorgabe und des Fahrermodells beschrieben.

1. Mastermodell

Das Mastermodell ist das Modell, das alle verteilt ablaufende Teilmodelle der Gesamtfahrzeugsimulation miteinander koppelt (Abbildung 3.4.1). Die Simulationsschrittweite beträgt 10 ms – die kleinste aller Schrittweiten bzw. Abtastraten in der Kommunikation zwischen den Teilmodellen. Mit ‚Data Exchange Server' wird das Subsystem mit den Kopplungen der Teilmodelle bezeichnet (Abbildung 3.4.2). Das Subsystem ‚Channel To Monitor' ist das Modell der Umsetzung der shared-memory-Kopplung, welche in diesem Beispiel das Mastermodell, daß auf einem Prozessor eines Doppelprozessorsystems abgearbeitet wird, mit dem Modell des Monitors koppelt, welches auf dem zweiten Prozessor abgearbeitet wird.

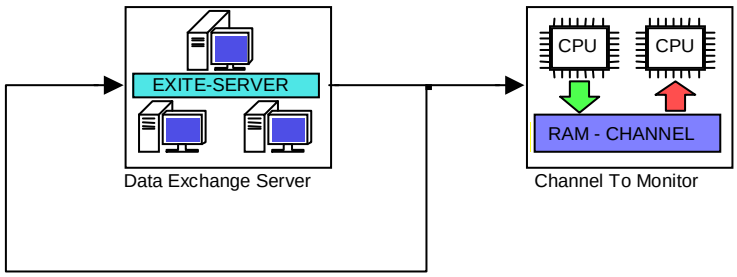


Abbildung 3.4.1: Das Mastermodell der vorliegenden Gesamtfahrzeugsimulation bestehend aus dem ‚Data Exchange Server' für die Kopplung der Teilmodelle und der shared-memory-Kopplung ‚Channel To Monitor' für das Übertragen der Schnittstellensignale zur Signalaufzeichnung.

Aus der Historie des Tools EXITE heraus begründet ist es ausschließlich möglich, Baumstrukturen als Topologien der Simulatorkopplung, das heißt Graphen ohne Schleifen, zu verwenden. Da aber alle Steuergeräte gleichrangig sind und diese untereinander kreuzweise kommunizieren können müssen, kann die Topologie der Gesamtfahrzeugsimulation nur ein sternförmiges Netz von Simulatoren sein. Es ist immer nur ein Stellvertretermodell (entspricht genau einer EXITE-Interfacedefinition) mit genau einem verteilten Simulator verbunden. Somit ergibt sich die in Abbildung 3.4.2 abgebildete Topologie. Hintergrund ist die Verwendung der EXITE-Version 1.3.4, welche ursprünglich auf CORBA beruhte (s. Kapitel 4.3, Punkt 1: Das EXITE-Interface).

Somit sind also alle Slaves, d.h. die Teilmodelle sternförmig mit dem Mastermodell verbunden (Abbildung 3.4.3), da der jeweilige EXITE-Master-Block, der sich in der Abbildung 3.4.2 in den weiß gezeichneten Subsystemen befindet, das Interface zum betreffenden Teilmodell in beide Richtungen realisiert. Das hat allerdings den Vorteil, daß an ***einer*** Stelle sämtliche Signale zwecks Aufzeichnung abgegriffen werden können. Über eine shared-

memory-Kopplung werden mehr als 900 CAN-Signale und physikalische Signale durch das Monitormodell in Simulationsschrittweite aufgezeichnet.

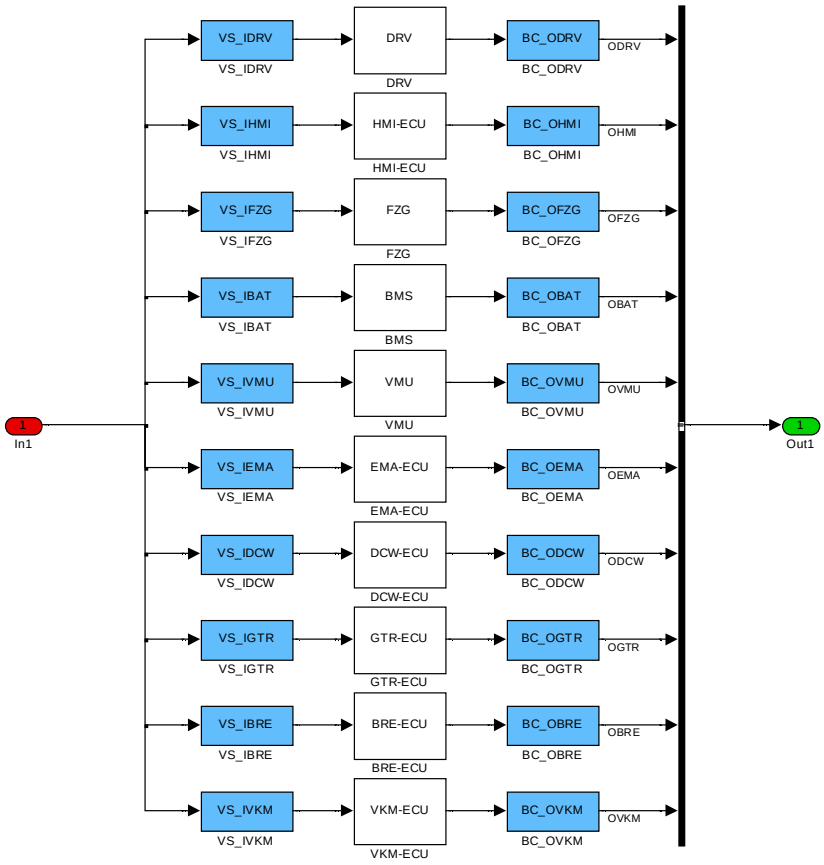


Abbildung 3.4.2: Aufbau des Data Exchange Servers im Mastermodell. Jedes verteilt simulierte Teilmodell hat hier einen entsprechenden Interfaceblock (weißes Subsystem), der als Output die Signale vom Teilmodell bereitstellt und als Input Signale zum Teilmodell weiterleitet. Die Subsysteme VS_Ixxx selektieren die zu sendenden Signale (Variablen Selektoren). Die Subsysteme BC_Oxxx fügen über sogenannte Bus Creatoren die empfangenen Signale dem Signalbus hinzu.

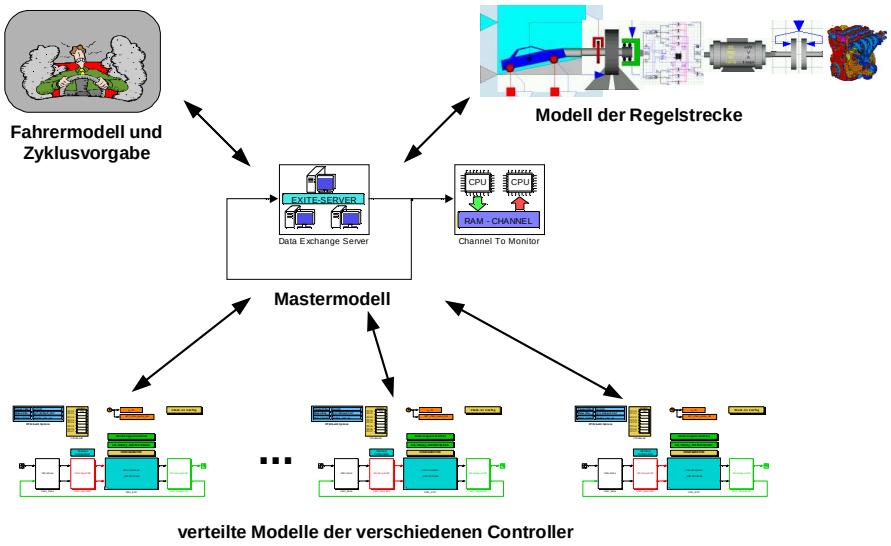


Abbildung 3.4.3: sternförmige Kopplung der verteilten Teilmodelle über das Mastermodell

Die Signale werden von allen Slaves auf einen gemeinsamen Bus gelegt (Ausgang Out1 in Abbildung 3.4.1), der verzögerungsfrei auf die Slaves zurückgeführt ist (Rückführung in Abbildung 3.4.1).
Durch die Variablenselektoren (Blöcke mit dem Präfix VS_ im Namen) werden den entsprechenden Slaves nur diejenigen Signale zugeführt, die die betreffenden Teilmodelle als Inputs auch nur erwarten. So werden algebraische Schleifen vermieden und die Schnittstellen zu den Slaves können plausibel schmal gehalten.

2. Monitor

Der Monitor oder besser das Monitormodell zeichnet sämtliche Schnittstellensignale aller Teilmodelle der vorliegenden Gesamtfahrzeugsimulation zum Zweck der offline-Auswertung bzw. des offline-Debuggings auf und wird deshalb wie das Mastermodell mit einer Schrittweite von 10 ms simuliert. Er ist ein eigener Teilnehmer im Simulationsverbund. Da der Monitor auf einem separaten Prozessor des Zwei-Prozessorsystems abgearbeitet wird, um die Kommunikation im Mastermodells, d.h. der Modellkopplung nicht zu behindern, ist dieser in einem separaten Simulinkmodell über eine shared-memory-Kopplung mit dem Mastermodell verbunden.

3. Fahrermodell und Fahrzyklusvorgabe DRV

Das Modell DRV liest einen vorher ausgewählten Geschwindigkeits-Zeit-Zyklus ein und generiert die Umweltgrößen für die Regelstrecke wie Steigung, Windkraft, Reibbeiwert des Straßenbelags usw. Außerdem erfolgt im Fahrermodell der Vergleich der Istfahrzeuggeschwindigkeit aus dem Regelstreckenmodell mit der Sollgeschwindigkeit aus dem Fahrzyklus. Diese Differenz wird einem PID-Regler zugeführt, der seinerseits eine Sollbeschleunigung berechnet. Über ein inverses Fahrzeugmodell wird diese Sollbeschleunigung in die Fahrervorgaben Bremspedal- und Fahrpedalwert überführt. Die Simulationsschrittweite des Fahrermodells beträgt 10 ms.

4 Aufbau und Handhabung der Clustersimulation

Ein sinnvoller, durchdachter Aufbau des Rechenclusters ist Grundvoraussetzung für eine leistungsfähige Simulation. Dies betrifft neben der Hardwarespezifikation und der eingesetzten Software auch das Zusammenspiel zwischen Hard- und Software. Aus diesem Grund soll der Aufbau der Gesamtfahrzeugsimulation im folgenden Kapitel genauer beschrieben werden.

In [Miegler et al. 2002, Miegler et al. 2003, Kovacevic 2002] wurden bereits eine vernetzte Simulation beschrieben. In der hier vorgestellten Gesamtfahrzeugsimulation wurde dieser Gedanke weiterentwickelt zu einer Fahrzeugsimulation hybridischer Antriebstränge, die auf einem Rechnersystem, bestehend aus 10 Doppelprozessorrechnern, verbunden mit einem doppeltem optischen Netzwerk, basiert. Dieses Rechnersystem wird nachfolgend auch Clusterrechner oder Rechencluster bezeichnet.

Um das in Kapitel 3 beschriebene komplexe System von Simulationen und Teilmodellen rechenbar zu machen, wird eine besonders leistungsfähige Hardware benötigt. Denn obwohl jeder hier verwendete Simulator primär für den Einzelplatz-PC entwickelt wurde und verfügbare PCs derzeit die Leistungsfähigkeit früherer Workstations besitzen, ist die in dieser Arbeit vorgestellte Antriebstrangsimulation auf einem Rechner nicht mehr (komfortabel) rechenbar.

Im folgenden Kapitel 4.1 wird auf den Hardwareaufbau des Clusterrechners eingegangen, da dieser entscheidend die Performance des Rechnerverbundes bestimmt. Jedoch auch die Art und Weise der Rechnerkopplung (Kommunikationshardware und -protokolle) und der Simulatorkopplung beeinflussen die Performance der Clustersimulation. Die Alternativen und die gewählten Lösungen sind in den Kapiteln 4.2 bzw. 4.3 beschrieben.
Welche Kommunikationsarten zwischen den Simulatoren möglich sind und welche für die hier vorliegende Gesamtfahrzeugsimulation letztendlich gewählt wurde, ist in Kapitel 4.4 dargestellt. Anschießend wird in Kapitel 4.5 beschrieben, wie der Aufbau einer Kommunikationsverbindung zwischen zwei Simulatoren abläuft.
In Kapitel 4.6 wird auf das Thema Synchronisation der Simulatoren und Determinismus und Reproduzierbarkeit der Simulation eingegangen.
Zum Schluß werden in Kapitel 4.7 Topologieaspekte der verteilten Modellierung systemtheoretisch betrachtet.

4.1 Aufbau des Clusterrechners

Im Rahmen der hier vorgestellten Gesamtfahrzeugsimulation wurde ein leistungsfähiger Clusterrechner, bestehend aus 10 Serverrechnern spezifiziert und in Betrieb genommen.

Abbildung 4.1.1: Photographie des Clusterrechners

Aufgrund der Anforderungen an die Rechengeschwindigkeit einer vernetzten Simulation müssen alle Komponenten, Hard- wie auch Software, aufeinander abgestimmt und optimiert sein. Im Folgenden wird kurz auf die Überlegungen, die zu dieser Hardware geführt haben, eingegangen. Sie spiegeln den Stand der Technik Ende des Jahres 2003 wider, als der Clusterrechner spezifiziert wurde.

1. Taktfrequenz und Anzahl der Prozessoren

Bei der Wahl der Prozessoren wurde von der Verfügbarkeit und der Unterstützung durch verbreitete Betriebssysteme ausgegangen. Randbedingungen waren:

- Ende 2003 waren Boards mit dem 64-bit AMD Athlon 64 FX-51 DDR-400 noch nicht verfügbar. Außerdem unterstützte zu diesem Zeitpunkt das Betriebssystem Windows-XP Professional nur 32-Bit-Architekturen.
- Der 32-Bit-Prozessor Intel Pentium 4 HT 3,2 GHz DDR-400 war laut [Computerbild 2003] der schnellste, verfügbare 32-Bit-Prozessor für PCs.
- Die Standardinstallation bei Volkswagen verwendet Windows-XP Professional als Betriebssystem.
- Linux wird nicht von der PC-Supportabteilung der Volkswagen Konzernforschung administriert, d.h. der Eigenaufwand durch Installation und Wartung würde bei dem Verwenden von Linux stark zunehmen.
- Der überwiegende Teil der in der Volkswagen Konzernforschung verfügbaren Programme ist nur unter Windows Betriebssystemen lauffähig.
- Es sollte mindestens ein Zweiprozessorsystem zur Anwendung kommen, um die Threads auf die verschiedenen Prozessoren zu verteilen. Idee: Ein Prozessor möglichst nur für die Berechnung eines Teilmodells und einen weiteren Prozessor für die Abarbeitung von Betriebssystemroutinen bzw. für die Kommunikation zwischen den Modellen.
- Windows-XP Professional Betriebssysteme unterstützen bis zu 8 Prozessoren, wobei kein Unterschied gemacht wird zwischen virtuellen und physikalischen Prozessoren

Aus diesem Grund wurde ein Serverboard mit zwei Prozessoren 3,2 GHz Intel XEON ausgewählt.
Im Folgenden verstehen sich sämtlichen Aussagen immer im Zusammenhang mit dem gewählten Betriebssystem Windows-XP Professional.

Der 3,2 GHz Intel XEON unterstützt das sogenannte Hyperthreading (HT©), d.h. ein realer Prozessor vom Typ Intel XEON verhält sich bei bestimmten Applikationen wie zwei virtuelle Prozessoren. Hyperthreading wird oft als virtuelles Zwei-Prozessorensystem ausgelobt. Diese Aussage stimmt nur bedingt.

Die CPU Intel XEON beinhaltet unter anderem[1]:

- 3 Integer-ALU (arithmetisch logic unit, d.h. Recheneinheit)
- 2 FPU_ADD (floating point unit for addition, subtraction and multiplying)
- 1 FPU_DIV (floating point unit for division, sine, cosine …)
- 2 FPU_STP (floating point unit mit bidirektionalem Zugriff auf Speicher)

D.h., daß bei einer Intel XEON CPU bei einfachen, logischen Operationen tatsächlich zwei Threads auf zwei Prozessoren aufgeteilt werden und parallel abgearbeitet werden, daß jedoch bei aufwendigen mathematischen Operationen die Parallelität verloren geht, da nur eine FPU_DIV zur Verfügung steht. Aus diesem Grund sollten maximal zwei parallel zu simulierende Teilmodelle auf einem Doppelprozessorrechner abgearbeitet werden.
Das Betriebssystem Windows XP Professional erkennt, durch das Hyperthreading hervorgerufen, vier Prozessoren bei Verwendung des Dual-Prozessor-Serverboards: zwei physikalische und zwei virtuelle.

1 Unter download.intel.com kann auf den FTP-Server von Intel zugegriffen werden. Indem man die die Ordnerstruktur durchblättert, kann unter /design/Pentium4/manuals/ auf eine große Anzahl von Handbüchern zugegriffen werden.

2. Größe und Schnelligkeit des Arbeitsspeicher

Um Engpässe in der Abarbeitung der Simulation zu vermeiden und weil zum Zeitpunkt der Spezifikation des Rechners das Konzept der Datenspeicherung nicht klar war, wurde von einem Arbeitsspeicher von mindestens 1 GB mit mindestens 333 MHz ausgegangen.

3. Taktfrequenz und Anzahl der Festplatte

IDE-Festplatten, die in den letzten zehn Jahren Standard waren, sollten durch die neuen, schnelleren Serial-ATA-Festplatten abgelöst werden. Serial-ATA-Festplatten waren Ende 2003 noch nicht verfügbar bzw. es gab auch noch keine Motherboards, die den seriellen ATA-Bus unterstützten. Aus diesem Grund wurden SCSI-Festplatten mit 10.000 U/min, 300 kbps Schreibgeschwindigkeit und Raid-System ausgewählt. Das Raid-System bedingt einen speziellen Controller, der zwei oder mehrere Festplatten verwaltet und Spiegelung der Festplatte (Raid 1) bzw. Verdopplung der Schreibgeschwindigkeit (Raid 0) realisiert. Eine Verdopplung der Schreibgeschwindigkeit wird erreicht durch Verwenden von physikalisch zwei Festplatten, indem der Datenstrom aufgeteilt und wechselweise auf beide Festplatten geschrieben wird. Raid 0 wurde bei allen Teilrechnern gewählt, damit auch das Schreiben auf die Festplatte nicht der Engpaß in der Performance der Simulation ist.

4. Grafikkarte

Wie in [Elektronik 2003] dargestellt, haben Grafikkarten einen entscheidenden Einfluß auf die Performance von Computersystemen, zumindest wenn aufwendige Darstellungen in Echtzeit berechnet werden. Da die visuelle Animation in der vorliegenden Gesamtfahrzeugsimulation eher ein Ausnahmefall ist, kann auf eine spezielle externe, den Prozessor entlastende Grafikkarte verzichtet werden. Die grafische Darstellung der Modelle in Simulink oder Dymola wird die Ressourcen der Prozessoren nicht merklich in Anspruch nehmen. Somit kann eine on-board Grafikkarte verwendet werden.

5. Taktfrequenz des Systembusses

Da die Taktfrequenz des Systembusses (FSB, engl.: Front Side Bus), der Bus zwischen Prozessor und RAM, die Schnelligkeit des Datenaustausches und damit der Programmabarbeitung bestimmt, muß auf einen maximal schnellen Systembus geachtet werden. Der FSB des verwendeten Serverboards wird mit 533 MHz getaktet.

6. Netzwerkkarte

Zwei Ethernetkarten sollten zur Verfügung stehen:

- 1 GBit/s Ethernet für die Verbindung aller Rechner in das Volkswagen Netzwerk
- 1 GBit/s Ethernet für die Emulation der Myrinet-Verbindung, solange die optischen Netzwerkkarten und die Treiber für Windows XP noch nicht verfügbar sind.

Da abzusehen war, daß die Kommunikation zwischen den Simulationsteilnehmern einen nicht unerheblichen Teil der Simulationszeit in Anspruch nehmen würde, sollte alternativ zum TCP/IP-Ethernet auch die Möglichkeit der Verwendung alternativer Netzwerkkarten sowie alternativer Kommunikationsprotokolle und Treiber geprüft werden. Wenn man in der Liste der Top 500 unter den weltweit leistungsstärksten Rechenclustern [TOP500] nachschaut, dann fällt auf, daß die Myrinet-Netzwerkkarte der Firma Myricom häufig Anwendung findet. Um eine hohe Bandbreite und eine kleine Latenzzeit bei der Datenüber-

tragung mit den Myrinet-Karten zu erreichen, wurde jedoch nicht die Version mit dem Kupferkabel als Kommunikationshardware verwendet, sondern die Karte M3F2-PCIXE-2 [Myricom 2003], welche über den PCI-X-Bus mit dem Server verbunden ist und ein doppeltes, optisches Netzwerk unterstützt: Im Gegensatz zu einem einfachen, optischen Netzwerk, wo die Netzwerkkarte je eine optische Leitung für die Sende- und Empfangsdaten hat, besitzt die Netzwerkkarte M3F2-PCIXE-2 je zwei optische Leitungen. Sie ist in der Lage, den Datenstrom aufzuspalten und parallel zu senden und erreicht somit nahezu eine doppelte Bandbreite gegenüber der einfach optischen Netzwerkkarte. Die Übertragung von Daten ab der Ebene Netzwerkkarte in Richtung optischer Hardware erfolgt echt parallel. Ab dem Zeitpunkt, wo die Daten auf dem PCI-X-Bus in Richtung Modell weitergereicht werden, ist die Datenübertragung nicht mehr parallel und belastet den Prozessor.

7. Taktfrequenz des PCI-Busses

Auch der PCI-Bus sollte mindestens so schnell sein, wie die Applikation, die auf diesem Bus aufsetzt. Im Fall der Gesamtfahrzeugsimulation war dies die Myrinet-Karte M3F2-PCIXE-2, die einen PCI-X-Bus mit 133 MHz benötigt.

Das Rechnerboard, welches die zwei mit 3,2 GHz getakteten Intel XEON CPU besitzt, konnte nicht völlig frei gewählt werden, da es auch alle eben genannten Punkte unterstützen sollte.
Das Serverboard X5DP8-G2, der Firma Supermicro Computer Inc., USA, [Supermicro 2004] entspricht diesen oben genannten Anforderungen.

4.2 Hardwareseitige Rechnerkopplung

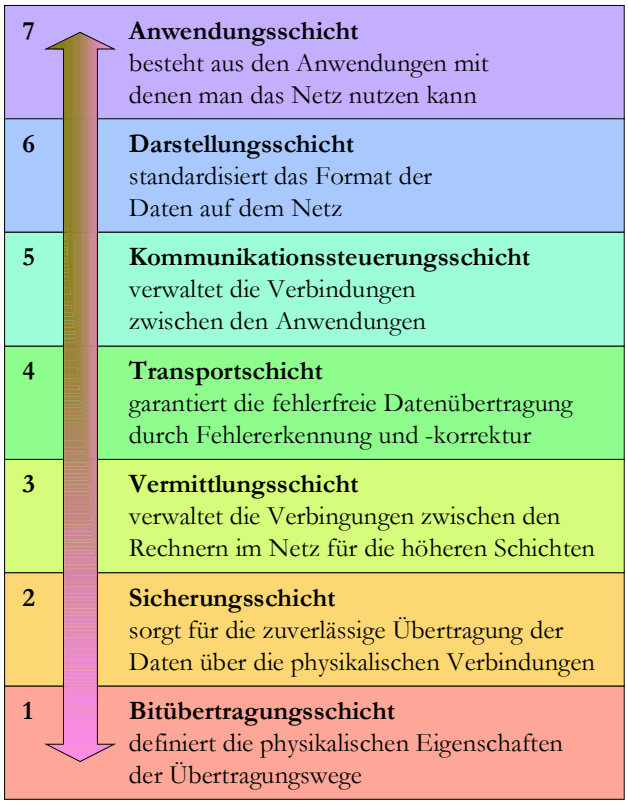
Jedes Teilmodell der Gesamtfahrzeugsimulation wird auf jeweils einem physikalischen Prozessor abgearbeitet. Die hardwareseitige Kopplung der 10 PCs untereinander erfolgt über wahlweise ein 1 Gbit/s Ethernet oder das optische Myrinet-Netzwerk. Softwareseitig sind die Simulatoren über die EXITE-API gekoppelt, die der obersten Schicht des ISO-OSI-Standdards (s. Abbildung 4.2.1.1) zuzuordnen ist.

Im Folgenden wird auf die unterschiedlichen Möglichkeiten der Kombination von Kommunikationsprotokollen mit der Kommunikationshardware eingegangen.

4.2.1 Kommunikationsprotokolle

Für die Beschreibung der Struktur und der Funktion von Protokollen der Datenkommunikation wird ein Architekturmodell zugrunde gelegt, welches von der International Standard Organization entwickelt wurde – das ISO-OSI-Schichtenmodell [Rothacker 1996].

Das ISO-OSI-Schichtenmodell besteht aus 7 Schichten (engl.: layer; Abbildung 4.2.1.1). Jeder einzelne Layer definiert Funktionen für die Datenkommunikation beim Austausch von Daten zwischen zwei oder mehreren Kommunikationspartnern.



(nach [Rothacker 1996])

Abbildung 4.2.1.1: Aufbau des ISO-OSI-Schichtenmodells

TCP/IP und MPI sind Beispiele für die Implementierung des ISO-OSI-Standards.

In den folgenden drei Kapiteln wird auf die Kommunikationsprotokolle CORBA, TCP/IP und MPI kurz eingegangen.

4.2.1.1 CORBA

CORBA, die Abkürzung für common object request broker architecture, ist eine objektorientierte, plattformunabhängige Kommunikationsarchitektur, die die Koordination, Synchronisation und Kommunikation zwischen Objekten realisiert. CORBA eignet sich sehr gut für die Realisierung der Infrastruktur der Middleware, den Schichten zwischen Anwenderprogramm und Betriebssystem, da es im Gegensatz zu beispielsweise C++ oder JAVA Mechanismen für die direkte Umsetzung der Interprozeßkommunikation bereitstellt. Grundlegende Funktionen sind die Erzeugung eines entfernten Objektes sowie Verbindungsaufbau zu diesem Objekt, der entfernte Methodenaufruf und die Übertragung großer Datenmengen. Die Objekt- und Slaveimplementierung sowie die Umsetzung und Anbindung an die EXITE API ist in [Bikker et al. 2002] ausführlich beschrieben.
Da die Kommunikation zwischen Objekten immer nach dem request/reply-Prinzip abläuft, d.h. es werden zum entfernten Objekt Daten gesendet und erst dann sendet dieses Objekt Daten zurück, sind mit dieser Middleware nur sequentielle und half-duplex Methoden der Kommunikation zwischen Simulatoren möglich (s. Kapitel 4.4).
Ein weiterer Nachteil von CORBA ist die Eigenschaft, daß die Kommunikation Client-Server-basiert ist. Das hat zur Folge, daß eine EXITE-Interfacebeschreibung immer nur ***eine*** Verbindung eines übergeordneten Modells (Master) zu einem Submodell (Slave) definiert und somit nur Baumstrukturen als Topologien der Simulatorkopplung, das heißt Graphen ohne Schleifen möglich sind. Außerdem wird mit jedem CORBA Request im Header des Datenpakets eine Object-ID versendet. Zu deren Identifikation müssen ORB Hash-Maps durchsucht werden, was Prozessorzeit beansprucht.
CORBA setzt im Allgemeinen auf das TCP/IP-Protokoll auf, d.h. es gestaltet als Middleware zwischen Applikation und Kommunikationsprotokoll die TCP-Aufrufe für den Anwender komfortabler.

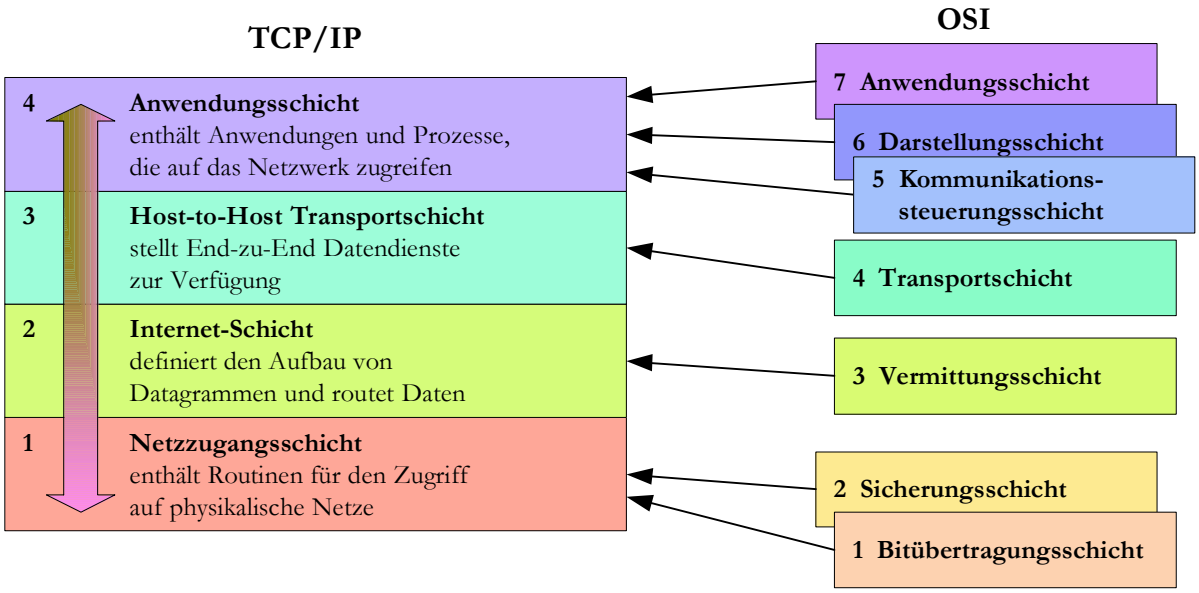
4.2.1.2 TCP/IP

TCP/IP ist der Standard-Protokoll-Stack mit folgenden Eigenschaften:

- Die Protokollspezifikationen ist herstellerunabhängig und frei zugänglich.
- Die Spezifikation ist von einem bestimmten Netzwerkmedium unabhängig.
- Es existiert eine standardisierte Schnittstelle zu Anwendungsprogrammen.
- Das Adressierungsschema ist fest definiert.

Im Gegensatz zum ISO-OSI-Basismodell mit sieben Schichten hat das TCP/IP-Schichtenmodell nur vier Schichten, die aber die sieben Schichten auf diese vier Schichten abbilden:

- Anwendungsschicht: Bereitstellung der Anwenderprogramme wie zum Beispiel FTP, Telnet, WWW etc.
- Transportschicht: Fehlererkennung und -korrektur durch das **Transmission Control Protocol (TCP)**
- Internet-Schicht: Definition der Adressierung und des Transportes von Daten über das **Internet Protocol (IP)**
- Netzzugangsschicht: Definition und Übertragung der Daten



(nach [Rothacker 1996])

Abbildung 4.2.1.2: Die Protokollarchitektur von TCP/IP im Vergleich zum OSI-Modell

Die Daten durchwandern die einzelnen Schichten beim Versenden von oben nach unten und bekommen von jeder weiteren, tiefer gelegenen Schicht einen weiteren Header mit Kontrollinformationen hinzugefügt. Dieser Prozeß wird mit Encapsulation (zu deutsch: Kapselung) bezeichnet. Wenn die Daten empfangen werden, müssen diese wieder die Schichten von unten nach oben durchlaufen und die Header werden wieder entfernt.

In den einzelnen Schichten werden die Daten wie folgt bezeichnet:

- in der Anwendungsschicht: Stream,
- in der Transportschicht: Segment,
- in der Internetschicht: Datagram und
- in der Netzzugangsschicht: Frame.

Die Daten werden als kleine, voneinander unabhängige Pakete, den Datagrammen, mit Kontrollinformationen im Header versendet. Es erfolgt kein direkter Verbindungsaufbau mit dem Zielrechner. Gehört die im Header kodierte Zieladresse zu einem Rechner im lokalen Netz, dann wird das Datagram direkt zugestellt. Im anderen Fall wird es an einen Router weitergeleitet, der aufgrund der Zieladresse das Datagram in das jeweilige Netz weiterleitet.

Während TCP/IP in heutigen Netzwerken einen bemerkenswerten Datendurchsatz erreicht, ist die Latenzzeit eher zu lang. Insbesondere folgende Eigenschaften kommerzieller TCP/IP-Implementierungen beeinflussen die Kommunikationsperformance:

(1) TCP/IP war für WAN-Netzwerke konzipiert
Auch wenn TCP/IP heute immer öfter in LAN-Netzwerken eingesetzt wird, ist dieses Protokoll nicht für diese Netze entwickelt worden. Zum Beispiel wird eine Checksumme in jedem übertragenen Frame für die end-to-end Zuverlässigkeit mitgesendet, obwohl durch die heutige Netzwerkhardware ohnehin ein CRC pro frame generiert wird. Das Generieren dieser Checksumme ist zeitintensiv und oft der Engpaß in der Paketverarbeitung. TCP/IP

ist Teil des (Betriebsystems-) Kernels und belastet somit den Prozessor. Das IP benutzt Header innerhalb des Frames, die ausschließlich für WAN-Umgebungen relevant sind. Des Weiteren benutzt das IP Eigenschaften, die in LAN-Umgebungen nicht sinnvoll sind, wie internetwork routing, in-flight packet fragmentation und reassambling. TCP/IP wurde konzipiert für die Kommunikation weitgehend autonomer Rechner, die minimal interagieren. Rechner in einem LAN-Netzwerk sind durch die Domäne oder die gemeinsam zu bearbeitende Aufgabe mit einander inhaltlich verbunden und kooperieren. Sie benutzen möglicherweise einen gemeinsamen File Service. (Im Fall der Gesamtfahrzeugsimulation ist der Masterrechner der Datenlogger und EXITE-Klassenserver.)

(2) TCP/IP besitzt ein aufwendiges Schichtenmodell
Standard Implementierungen des Socket-Interfaces und des TCP/IP-Protokolls teilen das Protokoll in mehrere Schichten. Das Socket-Interface ist normalerweise die oberste Schicht, welche über dem Protokoll sitzt. Die Protokollschicht kann Sub-Schichten enthalten. So sitzt der TCP-Protokollcode über dem IP-Protokollcode. Unterhalb der Protokollschicht ist der Interfacelayer, der mit der Netzwerkhardware kommuniziert. Auch der Interfacelayer hat normalerweise zwei Schichten: das network programming interface, welches die zu sendenden Daten vorbereitet, und den network device driver, welcher die Daten zur Netzwerkhardware sendet bzw. Daten von dieser empfängt.
Diese Implementierung erlaubt es, Protokollstacks für die verschiedensten Kombinationen von Protokollen, Programmierinterfaces und Netzwerkkarten zu bauen. Jedoch hat diese Flexibilität auch ihren Preis: Die Performance dieser Art Datenübertragung ist aufgrund des notwendigen Transports der Daten durch die vielen Schichten schlecht. Außerdem erfordert die Implementierung erhöhten Programmieraufwand. Jede Schicht kann einen eigenen Datenraum (Speicher) und eine eigene Codierung der Daten erfordern, was ein Transformieren der Daten bei Weiterleiten an die nächste Schicht erforderlich macht.

(3) TCP/IP benutzt eine komplizierte Speicherverwaltung
Heutige TCP/IP-Implementierungen benutzen komplizierte Mechanismen für die Speicherverwaltung aus verschiedenen Gründen: Einerseits bewirkt ein Multilayer-Protokollstack, daß die Header der Datenpakete addiert (oder gelöscht) werden müssen, wenn das Paket sich abwärts (oder nach oben) durch den Stack bewegt. Das Passieren der verschiedenen Schichten könnte einfach und schnell geschehen ohne exzessives Kopieren, wird aber bei heutigen Implementierungen nicht umgangen. Andererseits ist der Speicher innerhalb des Kernels eine rare Ressource, die in einem aus Betriebssystemsicht speicherplatzeffektiven Weg gehandhabt werden muß. Dieser zweite Grund war speziell richtig für ältere Systeme mit begrenztem Speicher.

Wenn die Anzahl der Schichten verringert würde, dann würden auch die eben genannten Nachteile wegfallen. Jüngste Entwicklungen zielen in diese Richtung. Wurde in der Vergangenheit vermehrt das Augenmerk auf eine Erhöhung der Bandbreite gesetzt, so wird jetzt mit neuen Implementierungen versucht, die Latenzzeit so weit wie möglich zu verringern.

4.2.1.3 MPI
MPI (engl.: Message Passing Interface) ist ein Standard [MPI 2005] der das Protokoll für die Kommunikation zwischen Workstations und bei Parallelrechnern definiert. Er wurde Anfang der neunziger Jahre des letzten Jahrhunderts in den USA definiert. Treiber sind weit verbreitet und teilweise frei verfügbar. MPI dient der nachrichtenbasierten Kommunikation bei paralleler Datenverarbeitung in heterogenen Rechnernetzwerken und stellt dem Programmierer verteilter paralleler Anwendungen eine standardisierte Schnittstelle (API) für die nachrichtenbasierte Kommunikation zur Verfügung. Es sind im Gegensatz zu

CORBA sowohl point-to-point Verbindungen als auch Kommunikationen zu Prozeßgruppen möglich. MPI ist wie CORBA plattformunabhängig und kann somit auf jedes beliebige System portiert werden.
Um die Anbindung an die Hardware zu gewährleisten wurde für MPI über Ethernet und MPI über GM/Myrinet die Library MPICH, Version 1.2.1 verwendet [MPICH 1998]. MPICH ist eine freie, portable Implementierung der vollen MPI-1-Spezifikation (genauer: MPI 1.2). Neben der MPI-Library enthält MPICH eine Programmierumgebung, um MPI-Applikationen zu entwickeln.
MPICH wird heute nicht weiter gepflegt. Eine neue Version einer freien MPI-Implementierung ist die Implementierung des MPI-2-Standards in der MPICH2-Library [MPICH2 2005].

4.2.1.4 GM-Treiber und Netzwerkkarte M3F2-PCIXE-2

GM ist ein proprietärer Treiber der Firma Myricom, Inc., USA der sich dadurch auszeichnet, daß der Overhead in der Prozessorbelastung, der durch die Kommunikation entsteht, minimiert wurde. Dabei wurde der Betriebssystemkernel durch die GM-API umgangen. Laut [GM 2005] ist die gemessene Belastung der CPU für das Senden und Empfangen einer Message kleiner 0,5 µs. Die Latenzzeit für 1KByte-lange Messages beträgt 10 µs und für Messages kürzer 200 Byte ist 5,05 µs. Diese Messungen wurden mit einem Dual-2,4-GHz Pentium4 XEON-Cluster durchgeführt und einem 64-bit, 133MHz PCI-X Slot. Allerdings handelt es sich bei diesen Tests um Kommunikationen zwischen nur zwei Rechnern. In einem Rechencluster sehen die Ergebnisse etwas anders aus (s. Kapitel 4.2.3). Im Rechencluster wurde die GM-Version 2.1.2 installiert.

Die Netzwerkkarte M3F2-PCIXE-2 ist wie GM auch ein Produkt der Firma Myricom, Inc. Diese Netzwerkkarte für optische Netzwerke hat nicht nur je einen optischen Adapter für Input und Output sondern je zwei, weshalb sie auch als doppelt-optische Netzwerkinterfacekarte bezeichnet wird. Diese Netzwerkkarte kann prinzipiell die Spitzen-PCI-Datenrate ausnutzen. Allerdings werden abhängig vom Systemspeicher und der PCI-Implementierung in der Realität niedrigere Datenraten erreicht (s. Anhang A).
Die Fiber-Ports der M3F2-PCIXE-2-Karte besitzen eine theoretische Bandbreite von 4 GBit/s. Die tatsächliche Bandbreite wird allerdings durch die Schnelligkeit der PCI-Karte, dem dynamischen Speicher und der PCI-Implementierung bestimmt. Die Hardwarelatenz zwischen Ports des LANai-Chips (s. Glossar) zweier Netzwerkkarten beträgt in Summe ca. 0,5 µs. Diese Latenzzeit entsteht durch das Konvertieren der Daten zum und vom optischen Layer. Die time-of-flight Latenz, also die Zeitdauer, die benötigt wird, um die Information über die optische Faser zu transportieren, beträgt ca. 0,0065 µs/m, also vernachlässigbare 0,013 µs bei einem 2 m Kabel im Fall der Gesamtfahrzeugsimulation.
Mit dem Befehl `gm_allsize` der GM-Library wurden folgende Werte für die Datenübertragung zwischen zwei Rechnern im vorliegenden Clusterrechner gemessen (s. Anhang B):

- minimale Latenzzeit: 5,8 µs
- maximale unidirektionale Bandbreite (Daten werden nur gesendet): 495 MB/s
- bidirektionale Bandbreite (gleichseitiges Senden und Empfangen von Daten): 815 MB/s.

4.2.2 Technologien der Rechnerkopplung

Über den in Kapitel 4.2.1 erwähnten Protokollen gibt es noch eine weitere Ebene: die Applikationsschicht. Hier werden die Modelle in Matlab/Simulink und Dymola bzw. als DLL miteinander auf Signalebene verbunden. Das Werkzeug, welches diese Verbindung auf Signalebene zwischen verschiedenen Modellierungstools und Modellen realisiert, ist das Tool EXITE der Firma Extessy AG in Wolfsburg [EXITE 2004]. Die EXITE-API stellt die Verbindung zwischen der Anwendung und den Kommunikationsschichten her.
In den folgenden Kapiteln werden die vier möglichen Rechner- bzw. Simulatorkopplungen beschrieben.

4.2.2.1 CORBA/TCP/IP über Ethernet

Grundsätzlich ist die EXITE-Kopplung auf einem TCP-basierten Netzwerk lauffähig. Das bekannte Ethernet-Kupferkabel ist dabei die Kommunikationshardware. Als Kommunikationssteuerschicht wird CORBA verwendet (Abbildung 4.2.2.1):

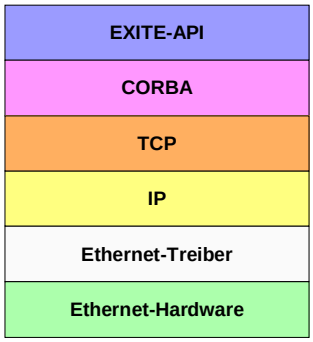


Abbildung 4.2.2.1: Schichtenmodell bei CORBA/TCP/IP über Ethernet

Im EXITE-Interface werden aufgrund von CORBA die Kommunikationsarten sequentiell und half-duplex (s. Kapitel 4.4) immer über dieses Schichtenmodell nach dem Prinzip request/reply realisiert (vgl. Kapitel 4.2.1.1). Da jedoch die Simulationsdauer der Gesamtfahrzeugsimulation bei der Anwendung dieser zwei Kommunikationsarten größer ist als bei full-duplex, schied diese Möglichkeit der Rechnerkopplung aus und wird nicht weiter betrachtet.

4.2.2.2 MPI/TCP/IP über Ethernet

Um die Zeit der Datenübertragung, die unter anderem durch die Kommunikationsart bestimmt ist, durch das Anwenden einer alternativen Middleware zu minimieren, wurde zusammen mit der Firma Extessy eine EXITE-Kopplung basierend auf MPI definiert und erarbeitet. MPI ersetzt dabei CORBA (Abbildung 4.2.2.2).

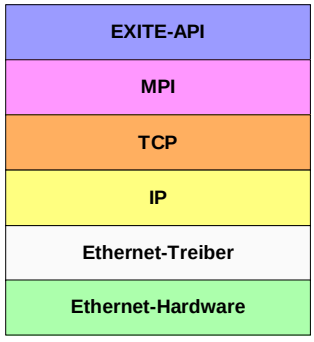


Abbildung 4.2.2.2: Schichtenmodell bei MPI/TCP/IP über Ethernet

Die Kommunikationsart full-duplex wird immer in Verbindung mit MPI als Kommunikationssteuerschicht verwendet.
Diese Kommunikationsart wurde während der Inbetriebnahme des Clusterrechners angewendet, als die GM-Treiber für Windows-XP noch nicht verfügbar waren.

4.2.2.3 MPI über Myrinet

MPI über Myrinet bezeichnet die Rechnerkopplung auf Basis der Softwareschnittstelle MPI über dem proprietären Treiber GM (s. Abbildung 4.2.2.3). Dieser Treiber wurde speziell für die Netzwerkkarten der Firma Myricom entwickelt, um Latenzzeiten zu minimieren. Da nur rudimentäre Sicherheitsalgorithmen implementiert sind, ist diese Art der Vernetzung grundsätzlich schneller als MPI/TCP/IP über Ethernet und ist damit die für die hier vorliegende Gesamtfahrzeugsimulation favorisierte Art der Rechnerkopplung.

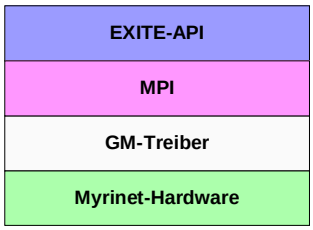


Abbildung 4.2.2.3: Schichtenmodell bei MPI/GM über Myrinet

4.2.2.4 CORBA/TCP/IP über Myrinet

Eine letzte mögliche Art der Vernetzung stellt CORBA/TCP/IP über Myrinet dar (s. Abbildung 4.2.2.4). Da TCP/IP Teil des Betriebssystems, d.h. des Kernels ist und aus diesem Grund ein Teil des GM-Treibers im sogenannten Kernel-Mode abgearbeitet werden muß, hat der GM-Treiber alle Rechte und kann die Schutzmechanismen des Betriebssystems umgehen und dieses im RAM zerstören. Ein Debuggen ist dann nicht mehr möglich und der Rechner muß neu gebootet werden.
Diese Variante dürfte sich hinsichtlich der Schnelligkeit ähnlich des CORBA/TCP/IP über Ethernet verhalten. Da beim Starten der Kommunikation die beteiligten Rechner abstürzten (sogenannter „Blue Screen“), schied diese Art der Vernetzung von vornherein aus.

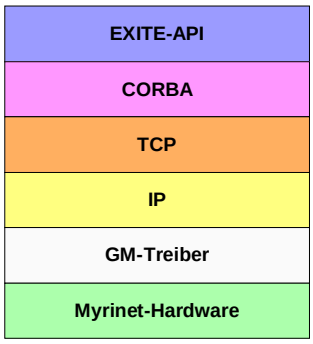


Abbildung 4.2.2.4: Schichtenmodell bei CORBA/TCP/IP über Myrinet

4.2.3 Bandbreite und Latenzzeiten

Bei der verteilten Simulation in einem Multiprozessorsystem bestimmen Bandbreite und Latenzzeit die Gesamtperformance der Simulation erheblich. Das Problem bei der Datenübertragung in der Gesamtfahrzeugsimulation stellt weniger die Bandbreite dar als die Latenzzeit. Die Bandbreite bestimmt die Fähigkeit eines bestimmten physikalischen Netzes, große Datenmengen in einer festen Zeit zu übertragen. Sie wird mit Mbit/s (oder auch Mbps), MB/s (Abk. für MByte/s) oder Gbps angeben. Die Latenzzeit ist die Codier- und Decodierzeit der Datenpakete. Sie hängt auch von dem verwendeten Kommunikationsprotokoll ab und verschlechtert insbesondere bei vielen kleinen Datenpaketen und fein gegliederten Tasks die Performance. Wenn man davon ausgeht, daß alle heutigen Netzwerke große Datenmengen problemlos übertragen können, dann hat die Latenzzeit aufgrund der hohen Abtastrate der Controller den größeren Einfluß auf die Performance der Simulation. In der Tabelle 4.2.3.1 sind zum Vergleich von Bandbreite und Latenzzeit beispielhaft verschiedene Vernetzungshardware bzw. Protokolle aufgelistet (Stand 2003). Die Vernetzungshardware und damit auch die Protokolle entwickeln sich derzeit rasant weiter. In http://www.dl.ac.uk/CFS/benchmarks/ werden high-end- und on-the-shelf-Hardware anhand anspruchsvoller Tests verglichen. Diese Tests werden laufend aktualisiert.

Hardware	Bandbreite in MByte/s	Latenzzeit in µs
Scali MPI (SCI Netzwerk)[1,2]	199,2	4,5
GM (Myrinet Netzwerk) [1,2]	245	7
SHMEM (SGI Origin 3800) [1,2]	1600	0,2 – 0,5
Firewire (nur Hardwarelevel)[1,2]	50	125
TCP/Ethernet [3]	12,5	160
Myrinet [3]	101	55
Myrinet 2000 [3]	240	20

Tabelle 4.2.3.1: Bandbreite und Latenzzeit einiger Multiprozessorarchitekturen

Wie schon in Kapitel 4.2.1.4 erwähnt wurde, haben einfache Laufzeitmessungen zwischen zwei Rechnern nicht die Aussage, wie sich ein Cluster unter Simulationsbedingungen verhält. Auf Basis der vorliegenden Antriebstrangsimulation wurden in einem Benchmark die Performance-Unterschiede von MPI/TCP/IP über Ethernet und MPI/GM über Myrinet untersucht (Tabelle 4.2.3.2).

		gemessene Simulationszeit tsim in s		Δt_{MyGM} [4] in ms
		MPI/TCP/IP über Ethernet	MPI/GM über Myrinet	
Simulations-Dauer Tsim in s	**10**	24,690948	23,266103	1,42
	100	243,57578	222,76587	2
	1000	2675,1208	2498,4615	1,77

Tabelle 4.2.3.2: Benchmark der in der Gesamtfahrzeugsimulation verwendeten Netzwerkverbindungen

1 Diese Werte wurden auf MPI-Softwarelevel gemessen inklusive des Overheads, der durch den API-Call im Fall des Sendens entsteht.

2 entnommen aus [Aronsson 2002]

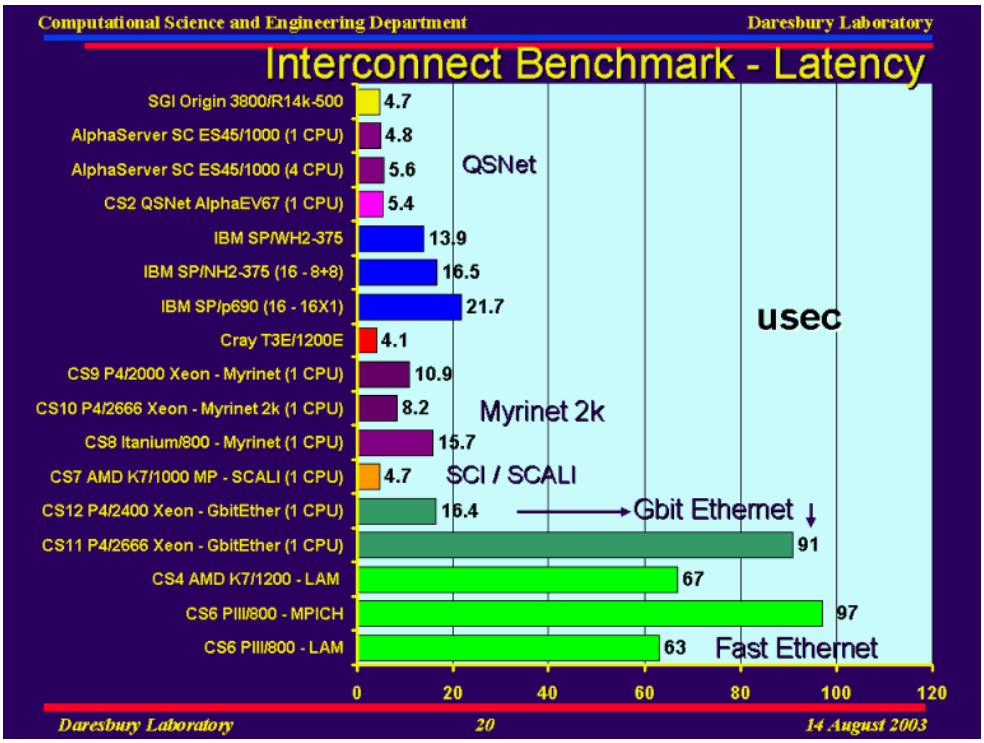
3 gemäß Untersuchungen des INRIA in Frankreich, Projekt-Team PARIS

4 Δt_{MyGM} ist der durchschnittliche Unterschied in der Latenzzeit beider Kommunikationsarten, der sich berechnet aus der Differenz der Simulationszeiten des Systems MPI/TCP/IP minus MPI/GM über Myrinet über Ethernet dividiert durch Tsim und Anzahl der Abtastungen pro Sekunde

Dazu wurde die Gesamtfahrzeugsimulation in unterschiedlichen Schrittweiten simuliert, um den Einfluß der Latenzzeit sichtbar zu machen. Die Konfiguration der Simulation entspricht der Simulation (9) in Tabelle 5.4.1. Aufgelistet sind die gemittelten Werte von jeweils 10 Messungen. In der Tendenz ist zu erkennen, daß MPI/GM über Myrinet schneller als MPI/TCP/IP über Ethernet ist. Da bei allen diesen Simulationen dasselbe Lösungsverfahren verwendet wurde, ist der Unterschied in den Simulationen nur aufgrund der unterschiedlichen Kommunikationsverfahren zu erklären. Man kann also ausgehend von der Tabelle 4.2.3.2 sagen, daß das Protokoll MPI/GM über Myrinet gegenüber MPI/TCP/IP über Ethernet ca. 1,8 ms pro Rechenschritt schneller ist. Das macht bei MPI/TCP/IP über Ethernet immerhin ca. 7 % der Rechenleistung eines Simulationsschrittes aus

Die Latenzzeit bei MPI/GM über Myrinet ist deshalb kleiner gegenüber MPI/TCP/IP über Ethernet, weil bei Myrinet unter Verwendung des proprietären GM-Treibers nicht in den Kernel-Mode des Prozessors umgeschaltet wird. Unter Umgehung von Sicherheitsroutinen des Betriebssystems (Kernel) und durch Weglassen von Schichten des ISO-OSI-Standards wird die Datenübertragung am Betriebssystem und am Standard vorbei schneller realisiert.

Zu den gleichen Ergebnissen kam eine englische Studie [Guest 2003] in einem aufwendigen Vergleich unterschiedlichster proprietärer (HP, Cray, SGI) und frei käuflicher (PC mit AMD-Athlon, Pentium III und IV) Systeme: Die Latenzzeit von Myrinet-Hardware lag unter 10 µs und die von Ethernet-basierter Hardware um 100 µs (s. Abbildung 4.2.3.1) Dabei wurden hochgradig parallele Simulationen auf den Gebieten der Molekularchemie, der Elektrophysik und der Werkstoffkunde durchgeführt.



(Quelle: [Guest 2003])

Abbildung 4.2.3.1: Benchmark der Latenzzeiten unterschiedlicher Rechencluster
Die Konfiguration CS10 und CS11 entspricht ungefähr der des in der Gesamtfahrzeugsimulation verwendeten Rechenclusters mit Pentium4/3,2GHz XEON. Der Aufbau der einzelnen Rechnerkonfigurationen ist in Anhang C beschrieben

Einer der schwerwiegendsten Gründe für lange Latenzzeiten in Netzen jeglicher Art ist das Warten auf die Verfügbarkeit des Mediums: So benutzt Ethernet traditionell CSMA/CD (carrier sense multiple access with collision detection). Dies bedeutet, daß der Ethernet-Treiber zu allererst prüft, ob das Medium gerade frei ist für die Datenübertragung. Wird das Medium benutzt, so wird die Datenübertragung zurückgestellt. Anderenfalls wird mit der Übertragung begonnen. Falls zwei oder mehrere Netzwerkkarten zur gleichen Zeit Daten übertragen möchten, wird eine Kollision erkannt. In diesem Fall wird die Übertragung des gegenwärtigen Frames durch den Ethernet-Treiber abgebrochen und es wird eine zufällige Zeitdauer gewartet, bevor wieder eine Datenübertragung begonnen werden kann. Um anderen Knoten im Netz die Chance zum Senden zu geben, wartet der Ethernet-Treiber 96 Bitübertragungszeiten nach jedem übertragenen Frame. Weiterhin besteht jeder Frame aus einem Header und den Nutzdaten. Allein der Header besteht aus mindestens 64 Byte. (24 Byte Netzwerk Header bestehend aus 8 Byte Präambel, 6 Byte Zieladresse, 6 Byte Quelladresse und 4 Byte CRC; 40 Bit TCP/IP kombinierter Header) Auch diese Headerdaten tragen zur Erhöhung der Latenzzeit bei.

Wie gezeigt wurde, hat MPI/GM über Myrinet gegenüber Standard-MPI/TCP/IP über Ethernet klare Vorteile. Dennoch gibt es seit einigen Jahren kommerzielle, proprietäre Lösungen für die Datenübertragung über Ethernet, die an die Performance spezieller Hardware heranreichen. Gründe dafür sind, daß das Ethernet eine kostengünstige Lösung ist und in Rechenclustern mit einem ***internen*** Kommunikationsnetzwerk, wie es bei dem verwendeten Cluster der Fall ist, die Wahrscheinlichkeit gering ist, daß Daten abhanden kommen und sogenannte Retransmissionen (Wiederholungen von Übertragungen) durchgeführt werden müssen. Daß heißt also, das Risiko von Datenverlusten ist klein und somit können auf die Sicherheitsschichten im Betriebssystem und im Transportprotokoll verzichtet werden. Weitere Gründe, die Kommunikation auf Basis von Ethernet zu optimieren sind unter anderem die weite Skalierbarkeit von 100 Mbps bis hin zu 80 Gbps, das große Angebot an Hardware wie Switches und Server, der Verzicht auf aufwendige optische Hardware sowie die Einfachheit in der Handhabung. Mehr als 30 der Top 100 der Rechencluster benutzen Ethernet als Kommunikationshardware.

So werden heute über proprietäre Ethernet-Lösungen Latenzzeiten von 5 µs bis hin zu 10 µs erreicht und machen somit der optischen Hardware Konkurrenz. Zu nennen sind dabei die Implementierungen LA-MPI des Los Alamos National Laboratory, Los Alamos, USA [Majumder et al. 2005] und SCSGigE der Firma Supercomputing Systems AG, Zürich in der Schweiz sowie Hardwarelösungen der Firma Foundry Networks, San Jose, Kalifornien, USA. Das Protokoll LA-MPI-TCP über Ethernet reicht in seiner Performance an LA-MPI-GM über Myrinet heran bzw. ist teilweise schneller
[Majumder et al. 2005].

4.3 Softwareseitige Simulatorkopplung

Die Parallelisierung der Antriebstrangsimulation durch die Simulation der einzelnen Teilmodelle verteilt auf mehreren Mikroprozessoren erfordert nicht nur die physikalische Kopplung der einzelnen Rechner und die damit verbundene Kopplung auf Ebene des Kommunikations- und Transportprotokolls sondern auch die Kopplung gleicher und unterschiedlicher Simulatoren, d.h. der Modellierungswerkzeuge, die die Simulation auf Anwenderebene durchführen.
Es gibt verschiedene kommerzielle Lösungen der Simulatorkopplung. In der hier vorliegenden Gesamtfahrzeugsimulation wird diese Simulatorkopplung, das Definieren der Schnittstellen zwischen den Teilmodellen und die Synchronisation der einzelnen Teilmodelle untereinander durch das Tool EXITE der Firma Extessy AG, Wolfsburg, gewährleistet. Ausschlaggebend für die Wahl von EXITE war unter anderem die bereits vorhandene Simulatorkopplung von Matlab/Simulink, ASCET-SD und ARTiSAN Real-Time Studio sowie die lokale Nähe der Extessy AG, wodurch eine intensive Zusammenarbeit und gemeinsame Weiterentwicklung von EXITE möglich war.

EXITE als Tool zur Simulatorkopplung kapselt die Kommunikation zwischen den Simulatoren. Der Anwender braucht sich somit keine Gedanken darüber machen, wie die Kommunikation auf der speziellen Rechnerhardware umgesetzt wird. EXITE setzt auf einer Schnittstelle im Modell des betreffenden Simulators auf und ist somit eine Modellschnittstelle. Im Gegensatz zu einer Programmierschnittstelle entbindet EXITE den Anwender von den Funktionsaufrufen der Netzwerk-API.

Die Umsetzung der Anbindung der einzelnen Simulatoren an EXITE ist in den folgenden Kapiteln beschrieben und kann im Detail [EXITE 2005] entnommen werden.

1. Das EXITE-Interface

EXITE bietet eine simulatorspezifische Schnittstellenbeschreibung, um verschiedene Simulatoren auf Schnittstellenebene zu koppeln. Das EXITE-Interface beschreibt die Eigenschaften für die Schnittstelle des betreffenden Simulationsteilnehmers, d.h. des zu simulierenden Teilmodells. Für jedes Teilmodell wird somit eine EXITE-Interfaceklasse angelegt. Die EXITE-Interfaceklassen werden zentral auf dem EXITE Cluster-Server (s. Glossar) als sogenannte Packages gespeichert und stehen allen Teilmodellen zur Verfügung.

In den EXITE-Versionen bis einschließlich Version 1.4.4 sind nur Baumstrukturen als Simulatorkopplungen möglich, also Grafen ohne Schleifen. Der Hintergrund liegt in der Historie von EXITE begründet, da EXITE ursprünglich nur für CORBA/TCP/IP über Ethernet als Standardkopplung von Simulatoren konzipiert war. Eine Eigenschaft von CORBA jedoch ist, daß immer nur ein Client mit jeweils einem Server kommunizieren kann. Somit beschreibt ein EXITE-Interface immer nur ***eine*** Verbindung eines übergeordneten Modells zu einem Submodell (s. Kapitel 4.2.2.1).
Ab EXITE-Version 2.0, die Ende 2005 verfügbar sein soll, können alle Teilmodelle unabhängig von der Dekomposition eines Gesamtmodells miteinander gekoppelt werden.

Da in der vorliegenden Clustersimulation mit der EXITE-Version 1.3.4 gearbeitet wurde, hat dies direkt Auswirkung auf die Topologie der Gesamtfahrzeugsimulation: In einem Netzwerk mit gleichrangigen Teilmodellen sind nur Sternstrukturen möglich (s. Abbildung 3.4.2 und 3.4.3).

Am linken Rand der Dialogbox des EXITE-Interfaceblocks (s. Abbildung 4.3.1) ist der EXITE-Interfaceklassenbrowser zu erkennen als Abbild des Repositories des EXITE-Cluster-Servers. Im EXITE-Interface sind die Anzahl der In- und Outputsignale, die von einem übergeordneten Modell (Master) zu einem Submodell (Slave) übertragen werden sollen, definiert. Weiterhin wird die Abtastzeit der Datenübertragung aller spezifizierten Input- und Outputsignale definiert. Da für eine Interfacebeschreibung nur jeweils eine Abtastzeit angegeben werden kann, muß bei mehreren Abtastzeiten in einem Teilmodell für jede Abtastzeit eine eigene Interfacebeschreibung angelegt werden. Weiterhin ist im Interface die Art der Datenübertragung, d.h. die Kommunikationsart (s. Kapitel 4.4) definiert sowie die Model-ID, eine alphanumerische Identifikationsnummer des Teilmodells.

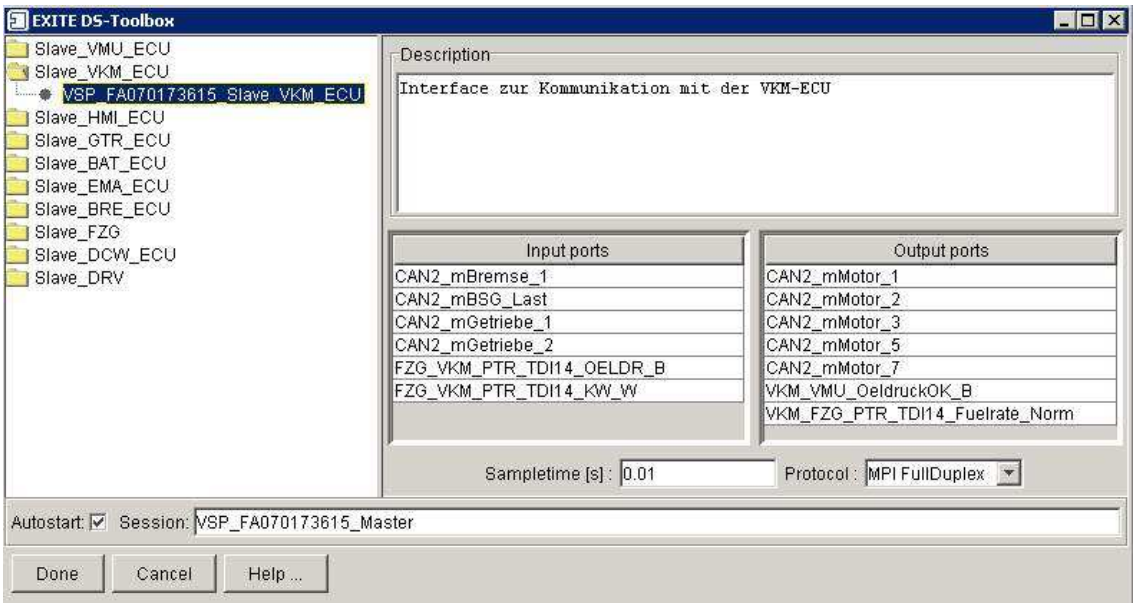


Abbildung 4.3.1: Die EXITE-Interfacebeschreibung am Beispiel des Interfaceblocks der VKM-ECU
Mit CAN2_... werden die entsprechenden CAN-Botschaften des Antriebs-CAN bezeichnet; Die Namen der übrigen Signale sind nach der Konvention <vonSender>_<nachEmpfänger>_<name> aufgebaut. Input ports bzw. Output ports beziehen sich aus Sicht des Slaves auf die Eingangs- bzw. Ausgangssignale des Servermodells des Verbrennungsmotorsteuergerätes.

2. *Kopplung von Simulinkmodellen*

Simulinkmodelle, die im Simulationsverbund verteilt simuliert werden sollen, müssen einen Simulink-EXITE-Interfaceblock enthalten, der mit Hilfe des Simulink-Librarybrowsers aus der EXITE-Library ausgewählt wird (Subsystem Slave in Abbildung 4.3.2). Mit diesem EXITE-Interfaceblock werden die Input- / Outputsignale, die das Teilmodell benötigt, verbunden.

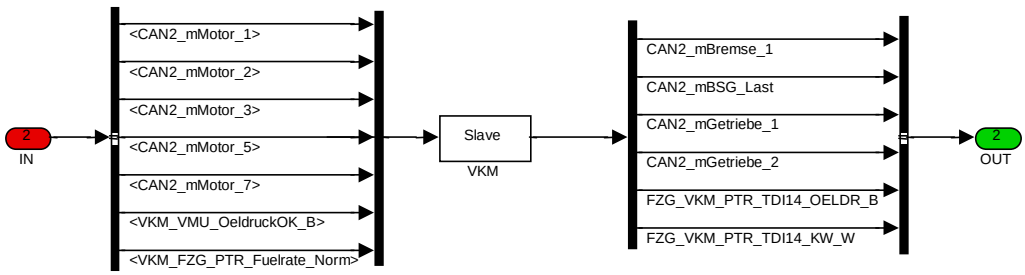


Abbildung 4.3.2: Beispiel für die Einbettung des EXITE-Slaveblocks im Simulink-Modell VKM-ECU. Bei Doppelklick auf das Subsystem Slave wird die in Abbildung 4.3.1 abgebildete Dialogbox geöffnet.

Das (Slave- oder Server-) Modell, welches seine Daten gemäß der Dekomposition an ein Teilmodell sendet und Daten von diesem erwartet, enthält einen sogenannten Masterblock als Simulink-EXITE-Interfaceblock (transparente Blöcke in der Abbildung 3.4.2). Das Teilmodell hat an der Stelle, wo es Daten vom übergeordneten Modell empfängt und Ergebnisse an das übergeordnete Modell sendet, den Slaveblock der Simulink-EXITE-Library. Die Dialogboxen der Interfacebeschreibungen in Master- und Slaveblock sind identisch.

3. Kopplung von durch Matlab-RTW generierten ausführbaren Dateien

Als Maßnahme zur Performancesteigerung soll die Gesamtfahrzeugsimulation auch mit kompilierten Matlab/Simulink-Modellen möglich sein. Das Kompilieren erfolgt durch den Matlab Real-Time Workshop (RTW), eine Toolbox von Matlab. Aus diesem Grund wurde gemeinsam mit der Firma Extessy eine RTW-exe-Kopplung definiert und erarbeitet.

Der Real-Time Workshop generiert aus dem grafischen Simulinkmodell compilerunabhängigen ANSI-C-Code. Dieser Code kann entweder für den PC in eine ausführbare Datei kompiliert werden, der sogenannten RTW-exe-Datei, oder mit Hilfe eines anderen Compilers für jede andere Hardware (z. Bsp. Mikrocontroller, DSP o. ä.) mit oder ohne Echtzeit-Betriebssystem kompiliert werden. Dadurch, daß das Modell bereits kompiliert ist und somit der Aufruf von Matlab/Simulink entfällt, beschleunigt diese Methode die Simulation. Im Fall der hier vorliegenden Gesamtfahrzeugsimulation wird die EXITE-Schnittstelle in die RTW-exe-Datei hineinkompiliert.

4. Kopplung mit Dymolamodellen

Es gibt zwei Möglichkeiten, Modelle, die in Dymola/Modelica erstellt wurden, mit anderen Teilmodellen im Simulationsverbund zu koppeln:

- Eine Simulink-s-function, die aus dem Dymolamodell heraus mit Hilfe einer speziellen Dymola-Lizenzoption generiert wird, eingebunden in ein Simulink-Teilmodell. Dieses Simulink-Teilmodell würde an die Gesamtfahrzeugsimulation über den eben beschriebenen Simulink-EXITE-Block gekoppelt.

- Das Generieren eines ausführbaren Files dynasim.exe aus dem Dymolamodell heraus, was durch die Standardlizenz von Dymola immer unterstützt wird, und das Anbinden an die Gesamtfahrzeugsimulation über spezielle EXITE-Dymola-Interfaces ist eine zweite Möglichkeit der Simulatorkopplung von Dymolamodellen.

Die erste Lösung ist sehr einfach und liegt auf der Hand. Allerdings würde durch das Starten einer zusätzlichen Matlab-Session ein unnötiger zeitlicher Overhead entstehen, da sämtliche Simulinkmodelle ansonsten als kompilierte, ausführbare Dateien aufgerufen werden. Ein weiterer Nachteil dieser Art der Kopplung besteht darin, daß dieses Simulink-Modell zur Verbesserung der Gesamtperformance trotzdem noch einmal mit dem Matlab Real-Time Workshop in eine ausführbare Datei kompiliert werden müßte. Bei Änderungen am Dymola Modell müssen dann zwei Kompilierungsvorgänge durchgeführt werden, was den gesamten Prozeß der Modellierung weniger handhabbar macht. Außerdem ist nicht klar, ob sich aus dem Dymolamodell überhaupt eine s-function generieren lassen würde: Nicht jede DAE läßt sich in eine ODE umwandeln, denn eine Randbedingung dieser Methode ist, daß der Matlab-Solver verwendet wird. Nur bei der RT-Option von Dymola (engl.: real time), die hier nicht zur Verfügung stand und das System unnötig unflexibel macht, kann in jedem Fall eine s-function mit einem Dymola-eigenen DAE-Solver generiert werden.

Im zweiten Fall existierte noch keine Dymola-Kopplung über EXITE. Diese Kopplung wurde gemeinsam mit der Firma Extessy definiert, erarbeitet und validiert und wird ab EXITE-Version 1.4.0 als Produkt der Firma Extessy kommerziell angeboten.

5. Kopplung mit selbsterstellter DLL

Da der in der Programmiersprache C entwickelte Algorithmus des Batteriemanagementsystems in Form von Quellfiles vorlag und es auch nahe lag, weitere Controller in Form von Black-Box-Modellen einbinden zu wollen, sollte auch die Möglichkeit gegeben sein, diese Controller in der verteilen Simulation zu simulieren. Diese Maßnahme kann auch als eine weitere Möglichkeit zur Performancesteigerung der Clustersimulation verstanden werden. Da diese Art der Simulatorkopplung durch EXITE nicht unterstützt wurde, wurde sie zusammen mit der Extessy AG definiert, die dann so in der Clustersimulation Anwendung fand.
Die Lösung ist, daß eine EXITE-API einschließlich der notwendigen Headerfiles beim Linken der Controller-Objektfiles dazugelinkt wird und somit eine ausführbare Datei erzeugt wird, die den Controller zusammen mit den Funktionsaufrufen der EXITE-API und damit der Anbindung an das Kommunikationsprotokoll enthält.
Somit konnte im Fall des Batteriemanagementsystems erreicht werden, daß der gesamte Funktionscode, der auch im realen Fahrzeug abgearbeitet wird, bitgenau und damit identisch in der Gesamtfahrzeugsimulation abläuft. Möglich ist das nur bei Integercode und wenn eine klare Trennung zwischen Funktionscode und prozessorspezifischer Basissoftware gemäß Autosar besteht.

Ab EXITE-Version 2.0 soll diese EXITE-API kommerziell verfügbar sein.

4.4 Kommunikationsarten zwischen den Simulatoren

Es gibt drei alternative Möglichkeiten für die Art und Weise der Datenübertragung zwischen Simulatoren, die im EXITE-Interfaceblock eingestellt werden können [EXITE 2004]:

- **sequentiell:** In einer verteilten Simulation wird zu jedem Zeitpunkt immer nur genau ein Teilmodell abgearbeitet. Die jeweils anderen Teilmodelle warten solange, bis das das in Bearbeitung befindliche Teilsystem seinerseits die Abarbeitung im betreffenden Abtastzeitschritt beendet hat und seine Ergebnisse über das Interface den anderen Teilmodellen zur Verfügung stellt (Abbildung 4.4.1).

- **half-duplex:** Die Simulationen der Teilmodelle werden parallel über die jeweiligen einzelnen Threads durchgeführt, d.h. quasiparallel bei Threads auf dem gleichen Prozessor oder echt parallel bei Threads auf verschiedenen Prozessoren. Die Kommunikation zwischen den Teilmodellen, d.h. das Senden und Empfangen erfolgt sequentiell (Abbildung 4.4.2).

- **full-duplex:** Die Simulationen der Teilmodelle werden parallel über die jeweiligen einzelnen Threads durchgeführt, d.h. quasiparallel bei Threads auf dem gleichen Prozessor oder echt parallel bei Threads auf verschiedenen Prozessoren. Das Senden und Empfangen erfolgt gleichzeitig, wenn dies die Netzwerkkarte unterstützt und kann auch parallel zur Simulation erfolgen (Abbildungen 4.4.3). Daraus folgt, daß das hinsichtlich der Kommunikation plus Algorithmus aufwendigste Modell die Performance des Gesamtsimulationsverbundes bestimmt.

Im Fall der hier vorliegenden Gesamtfahrzeugsimulation mit Rechnern mit Mehrprozessorsystemen wurde die full-duplex-Methode als Übertragungsmethode gewählt, weil sie die Möglichkeiten des Systems voll ausschöpfen kann und somit die besten Voraussetzungen bietet für ein leistungsfähiges Simulationssystem.

Im Folgenden werden die verschiedenen Arten der Datenübertragung noch einmal bildlich dargestellt.
Dabei bedeuten:

Algorithmus 1 (nT)	Berechnung des Algorithmus 1 zum Zeitpunkt $t + n * T_{ab}$[1]
Senden	Prozessorzeit für das Senden von Daten
Empf.	Prozessorzeit für das Empfangen von Daten

1 Abtastzeit bzw. Umlaufzeit des Algorithmus

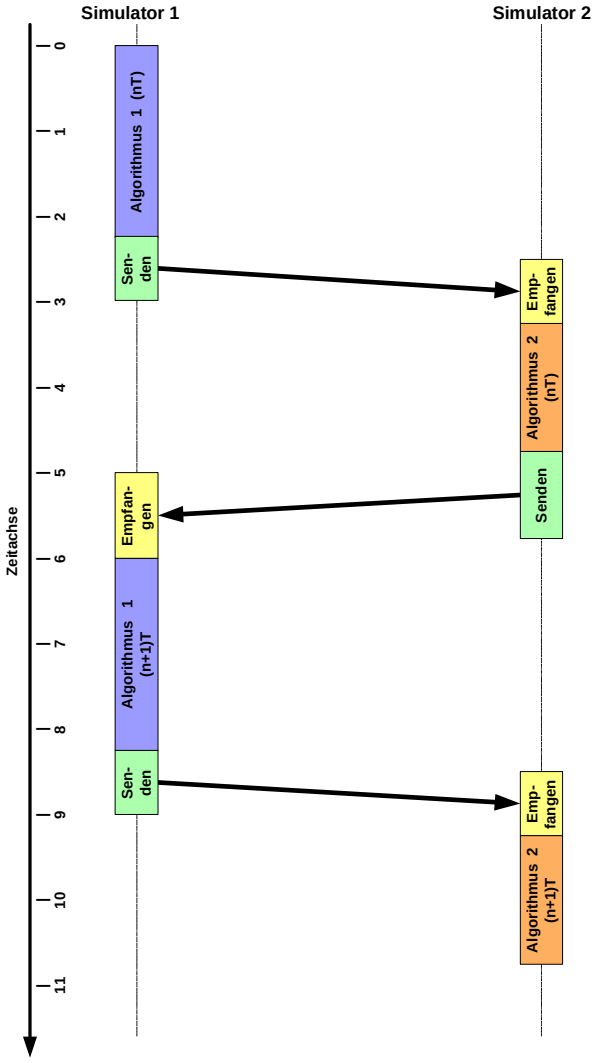


Abbildung 4.4.1: Prozessorbelastung und zeitlicher Ablauf bei der sequentiellen Kommunikation
Auf der Zeitachse sind virtuelle Zeiteinheiten dargestellt zur Veranschaulichung der einzelnen Prozeßschritte der Kommunikation. Jeweils gleiche Prozesse dauern die gleiche virtuelle Zeit.

Zur Erklärung zu den Abbildungen 4.4.1 bis 4.4.3:
Kurze Zeit später, nachdem der Prozessor des Simulators 1 die zu sendenden Daten an die Netzwerkkarte weitergeleitet hat und diese die Daten versendet hat, erfolgt das Decodieren der Daten auf der Empfängerseite durch die Netzwerkkarte bzw. durch die verschiedenen Schichten des Netzwerkprotokolls und dann das Entgegennehmen der Daten durch den Prozessor des zweiten Simulatoren. (In den Abbildungen 4.4.1, 4.4.2 und 4.4.3 ist jeweils eine Dauer von ca. 0,25 Zeiteinheiten angenommen, was den Versatz von Senden und Empfangen von ca. 0,25 Zeiteinheiten erklärt.)
Die Netzwerkkarten arbeiten somit im Hintergrund. In den Abbildungen 4.4.1, 4.4.2 und 4.4.3 soll die reine Prozessorbelastung dargestellt werden unter der Annahme, daß jeder Simulator auf einem eigenen PC abgearbeitet wird.

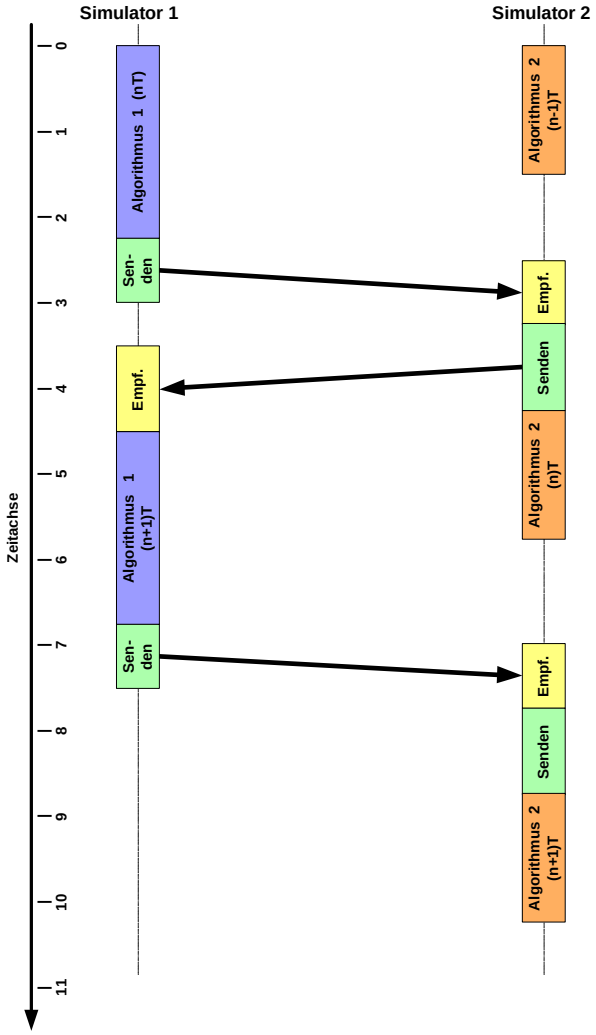


Abbildung 4.4.2 Prozessorbelastung und zeitlicher Ablauf bei der half-duplex Kommunikation

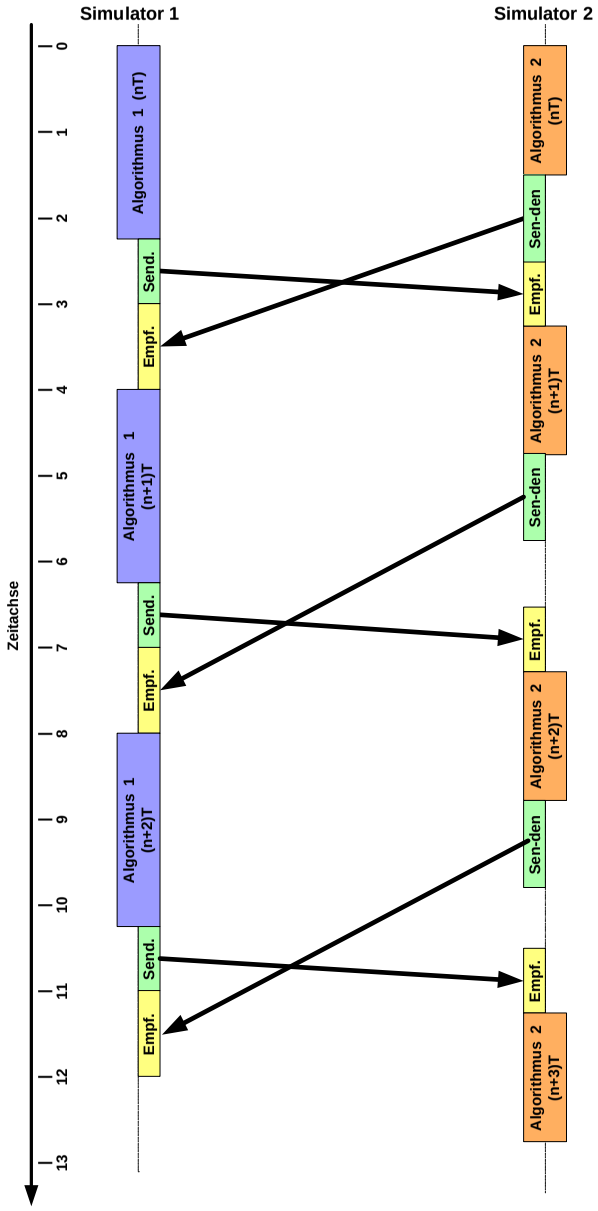


Abbildung 4.4.3: Prozessorbelastung und zeitlicher Ablauf bei der full-duplex Kommunikation

4.5 Aufbau der Kommunikation zwischen den Simulatoren

Die Kommunikation zwischen den einzelnen Simulationspartnern erfordert eine eindeutige Zuordnung der Teilmodelle zu dem zu simulierenden Gesamtmodell. Dies erfolgt über die sogenannte Model-ID, die im Modell gespeichert ist.

Wie schon oben erwähnt, werden auf einem dedizierten Rechner, dem sogenannten EXITE Cluster-Server, alle Interfaceklassen der Teilmodelle abgespeichert, d.h. die EXITE-Interfaces und die Verbindungsdaten (Abtastzeit, Sessionname – ein Zusammengehörigkeitsmerkmal aller vorhandenen Modelle zu einer Simulationssession, Kommunikationsart usw.).

Die EXITE-Installationen werden durch ein EXITE-Propertyfile so konfiguriert, daß sie diesen EXITE Cluster-Service nutzen können. Auf den Rechnern der Teilmodelle ist der EXITE Model-Service gestartet (s. Glossar). Im Mastermodell ist der EXITE-Masterblock stellvertretend für die einzelnen Teilmodelle eingefügt. Dieser Block wurde über den Simulink-Library-Browser aus der EXITE-Library per Drag & Drop ausgewählt. Auf dieselbe Art wurde in allen verteilten Teilmodellen der EXITE-Slaveblock anstelle der Verbindung zum Mastermodell aus der EXITE-Library in Simulink per Drag & Drop eingefügt. Wenn alle Einträge im Klasseninterface gemacht wurden, kann das jeweilige Teilmodell auf dem verteilten Rechner eingecheckt werden, d.h. das aktuelle Simulink-Modell bzw. das entsprechende mdl-File wird an einen dem lokalen EXITE Model-Server bekannten Ort auf der Festplatte gespeichert.

Anmerkung: In allen Modellen außer dem Mastermodell wurde die Startart Autostart im Gegensatz zu ‚manuell' gewählt. Das hat zur Folge, daß durch EXITE die Teilmodelle automatisch gestartet werden, sobald sich das Mastermodell beim EXITE Cluster-Service angemeldet hat. Im Gegensatz dazu können auch einzelne Teilmodelle per Hand gestartet werden. In diesem Fall wartet der gesamte Rechencluster bis sich das letzte Teilmodell beim EXITE Cluster-Service angemeldet hat. Da diese zweite Variante aufgrund der Vielzahl der Teilmodelle weniger praktikabel und nur im Fall des manuellen Debuggens von Teilmodellen sinnvoll ist, wird im Nachfolgenden immer die Startart Autostart vorausgesetzt.

Wenn das Mastermodell gestartet wird, dann meldet jeder EXITE-Stellvertreterblock (EXITE-Masterblock der Teilmodelle) eine Model-ID beim zentralen EXITE Cluster-Service an mit dem Wunsch nach einem ferngesteuerten Autostart ((1) in Abbildung 4.5.1). Da auf jedem Simulationsrechner der EXITE Model-Service gestartet ist, wird dieser nun vom EXITE Cluster-Service dazu veranlaßt, das der Model-ID entsprechende Teilmodell zu starten, sobald der EXITE Cluster-Service den entsprechenden Simulationsrechner gefunden hat ((2) in Abbildung 4.5.1). Das Mastermodell wird solange blockiert, wie die Model-ID nicht zugeordnet werden kann. Wird nun das Teilmodell (Slave-Modell) der Gesamtfahrzeugsimulation als paralleler Prozeß gestartet, dann meldet es sich bei demselben EXITE-Service mit seiner Model-ID an ((3) in Abbildung 4.5.1). Nun wird das Mastermodell dem Slave-Modell als sein Kommunikationspartner zugewiesen und der Datenaustausch kann nach der Definition gemäß dem EXITE-Interface (zum Beispiel halfduplex, full-duplex) stattfinden ((4) in Abbildung 4.5.1).
Wenn mehrere geeignete Teilmodelle ein und derselben Model-ID im Verbund existieren, dann wird das Teilmodell zuerst gestartet, welches der EXITE Cluster-Service zuerst findet [EXITE 2004].

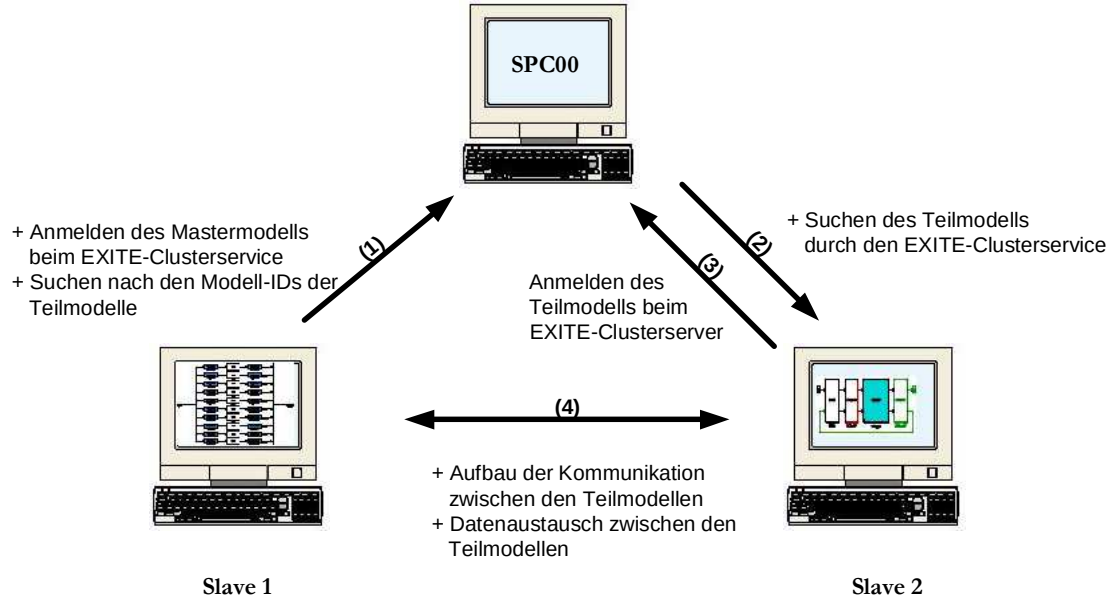


Abbildung 4.5.1: Aufbau der Kommunikation zwischen den Teilmodellen. Der Simulationsteilnehmer am oberen Rand ist der Masterrechner SPC00, auf dem zugleich der EXITE Cluster-Server installiert ist. Die unteren beiden Simulationsteilnehmer sind beliebige Slaves.

Aufgrund der MAC-Adresse (engl.: Media Access Control) im Fall der Netzwerkkarte oder der IP-Adresse im Falle eines Switches, wird das Datenpaket direkt an die richtige Netzwerkkarte gesendet. Die Auflösung der Clientnamen auf IP-Ebene findet nur beim Start der Simulation statt und hat somit keinen Einfluß auf die Performance der Gesamtfahrzeugsimulation, da die Clientnamen im Cache gehalten werden.

Dieser eben beschriebene Aufbau der Kommunikation in Verbindung mit der im folgenden Kapitel beschriebenen priorisierten Abarbeitungsreihenfolge durch Matlab/Simulink bestimmt somit die Synchronisation aller Teilnehmer zum Startzeitpunkt.

4.6 Reproduzierbarkeit und Synchronisation der Clustersimulationen

Eine Simulation liefert reproduzierbare Ergebnisse, wenn sie sich mit identischen Ergebnissen wiederholen läßt. Dies ist bei der hier vorgestellten Methodik nachweislich erfüllt, wenn eine Simulation ohne Veränderungen ein zweites Mal gestartet wird. Die Ergebnisse beider Simulationen sind in allen Signalen zu allen Abtastzeitpunkten identisch. Allerdings bedeuten schon kleine Änderungen auch nur in einzelnen Simulatoren Abweichungen im Kleinsignalbereich: Es kommt zu kleinen Unterschieden in den Amplituden und Phasen von Signalen aufgrund der Sensitivität von Reglern, sobald die Eingangssignale nicht mehr identisch mit denen der Ausgangssimulation sind, je nachdem wie umfassend die Abweichungen sind.

Die Synchronisation der verteilten Simulationen der Teilmodelle der Gesamtfahrzeugsimulation wird zum einen über den im letzten Kapitel beschriebenen Verbindungsaufbau zwischen den Simulationsteilnehmern durch das Tool EXITE sichergestellt. Der theoretische Ansatz ist in [Bikker et al. 2002] beschrieben. Historisch bedingt wurde in den EXITE-Interfaceblock eine Zeitverzögerung hineinprogrammiert, um bei sprungfähigen Systemen algebraische Schleifen zu verhindern. Ausgangspunkt der Entwicklungen waren Modelle in Matlab/Simulink. Der Abarbeitungszustand der einzelnen Teilmodelle kann sich EXITE-bedingt somit höchstens um einen Abarbeitungsschritt unterscheiden.
Zum anderen wird die Synchronisation aller Teilmodelle über das Mastermodell durch die priorisierte Abarbeitungsreihenfolge in Simulink sichergestellt. So werden durch Matlab/Simulink die Simulink-Blöcke der Stellvertreter gemäß ihrer alphabetischen Reihenfolge priorisiert abgearbeitet: Wird vor die Namen der Stellvertreterblöcke im Simulink-Mastermodell gemäß Abbildung 3.4.2 ein anderer Buchstabe gesetzt, so werden die Blöcke durch Matlab/Simulink neu und in anderer Reihenfolge priorisiert, so daß sich eine andere Abarbeitungsreihenfolge der Teilmodelle ergibt und somit eine leicht unterschiedliche Simulation im Kleinsignalbereich. Tendenziell liefern solche Simulationen vergleichbare Ergebnisse. Eine einmal eingangs durch Matlab/Simulink vergebene Abarbeitungsreihenfolge aufgrund der Priorisierung bleibt, keine Änderung am Modell vorausgesetzt, für alle Zeiten bestehen.
Allerdings auch in der Realität ergibt sich aufgrund des parallelen, durch die Steuergeräte selbst gesteuerten Hochlaufs eine rein zufällige zeitliche Abfolge der Kommunikation zwischen den Steuergeräten, die sich zum einen während einer Fahrt (‚Fahrt' ist gleichzusetzen mit ***einer*** Simulation.) ändern kann und zum anderen von Fahrt zu Fahrt unterschiedlich sein wird. Laufzeitunterschiede auf dem CAN bewirken Jittern und Verschieben der zeitlichen Zusammenhänge. (Der CAN-Bus arbeitet prinzipiell asynchron.)
Das heißt also einen Determinismus, welches Steuergerät in zeitlicher Abfolge genau mit welchem Steuergerät kommuniziert, gibt es in der realen Welt nicht und braucht es somit auch in der Simulation notwendigerweise nicht geben. Etwas anderes wäre es, wenn anstatt des asynchronen CAN eine Kommunikation basierend auf einem zeitgesteuerten Protokoll, z. Bsp. TTP-CAN genutzt wird (TTP steht für time triggered protocol, s. [TTP 2005, Kopetz et al. 1992, Kopetz et al. 1993]), wo innerhalb bestimmter Zeiten, eine Reaktion von einem Steuergerät erwartet wird. TTP findet Anwendung in sicherheitskritischen Systemen wie den by-wire-Systemen steer-by-wire oder der elektromechanischen Radbremse bei Kraftfahrzeugen.

Im Folgenden wurde ein Modell gemäß der Abbildung 4.6.1 für die Untersuchung der Synchronisation herangezogen, in welchem zwei Slaves über einen Master miteinander kommunizieren, also genau so, wie in der Clustersimulation die Kommunikation zweier Controller realisiert ist. Das Ergebnis ist in Tabelle 4.6.1 bzw. in der Abbildung 4.6.2 zu sehen.

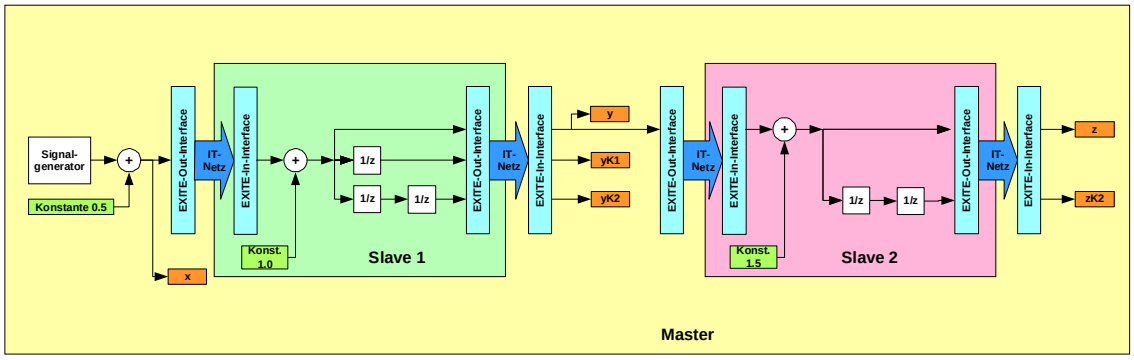


Abbildung 4.6.1: Beispiel für die Untersuchung des Determinismus der Simulation anhand eines Masters und zweier Slaves. Dabei sollen Daten aus dem Master an Slave 1 übergeben werden und von Slave 1 an den Slave 2. Dies entspricht einer typischen Signalkette. Die relevanten Daten, die vermessen werden sollen, sind in den orangen Kästchen mit x, y, yK1, yK2, z und zK2 bezeichnet. Dabei bedeuten K1 und K2 die Verzögerung um eine bzw. zwei Abtastzeiten. Alle Simulatoren und auch die Kommunikation besitzen die gleiche Schrittweite bzw. Abtastzeit (10 ms). Die Simulation wurde in Matlab/Simulink programmiert.

Bei Schrittweiten ist zu unterscheiden zwischen der Schrittweite in der Modellierung einzelner Teilmodelle und der im EXITE-Interface eingestellten Abtastzeit der Kommunikation. Im Beispiel gemäß Abbildung 4.6.1 wurde wie in der Gesamtfahrzeugsimulation die Simulationsschrittweite der Abtastzeit der Kommunikation gleichgesetzt, nämlich 10 ms. Zwischen den Signalen in Abbildung 4.6.1 ergibt sich folgende Verzögerung:

Signal a	Signal b	Verzögerung in Simulationsschrittweiten des Mastermodells zwischen dem Signal b zu dem Signal a	Verzögerung in Simulationsschrittweiten des Mastermodells zwischen dem Signal b zum Ausgangssignal x
x	y	2	2
y	yK1	0	2
yK1	yK2	1	3
yK2	z	0	3
z	zK2	1	4

Tabelle 4.6.1: Verzögerung zwischen den einzelnen Signalen in dem Beispiel gemäß Abbildung 4.6.1

Wie aus der Tabelle 4.6.1 ersichtlich, wird jedes durch den Controlleralgorithmus selbst nicht verzögerte Signal (im Beispiel Signal y und z) genauso wie die um 1/z verzögerten Signale um eine Schrittweite verzögert. Dies entspricht einem bei digitalen Abtastsystemen notwendigen Abtast- und Halteglied. Aus diesem Grund und auch aufgrund einer gewissen Systematik sollte bei der Modellierung in Matlab/Simulink in einer Signalvorverarbeitung jedes Signal im Simulink-Controller per se um 1/z verzögert werden.

Die in der Abbildung 4.6.2 zum Zeitpunkt t = 2s dargestellten Zeitverhältnisse bleiben für alle Zeiten konstant und sind von Simulation zu Simulation reproduzierbar.
Die Verzögerung zwischen zwei virtuellen Steuergeräten in der Gesamtfahrzeugsimulation entspricht der Verzögerung zwischen Slave 1 und Slave 2 und beträgt eine Abtastzeit in der Kommunikation. Dies entspricht genau einer maximalen CAN-Verzögerung eines Signals mit CAN-Wiederholrate gleich Abtastrate der Kommunikation dieser SIL.

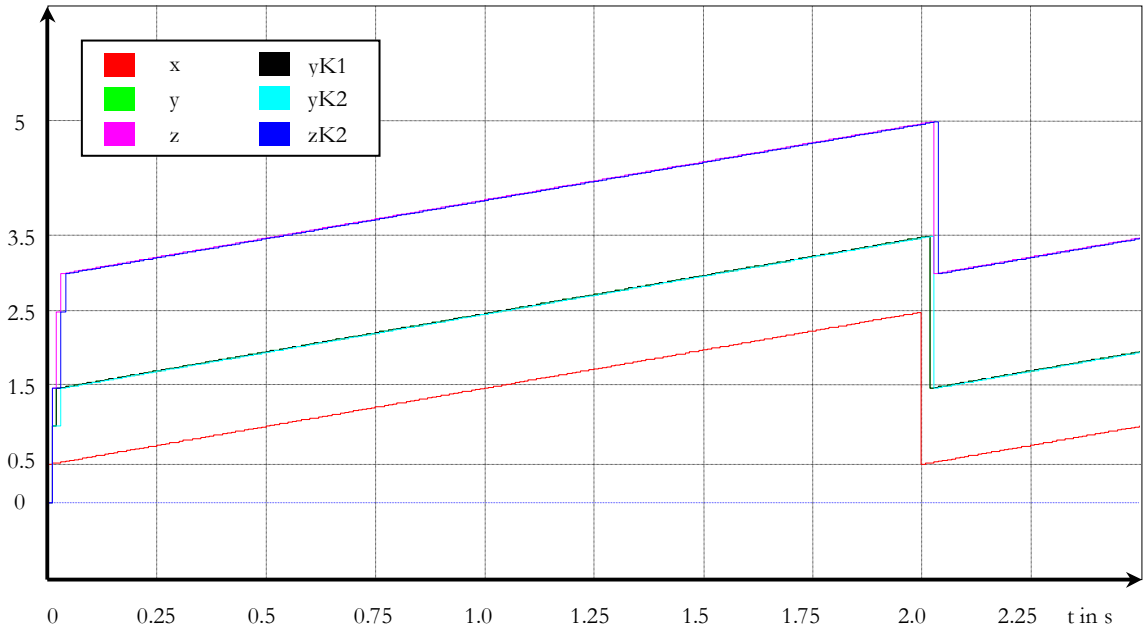


Abbildung 4.6.2: Vermessung einer Simulation gemäß Abbildung 4.6.1
Zu erkennen ist die Rasterung der Signale in Simulationsschrittweite von 10 ms. Als Signalgenerator dient ein Sägezahngenerator mit Amplitude 2 und Periode 2s. Nach einer Startphase wiederholt sich in Periodenabstand die zeitliche Abfolge aller Signale gemäß Zeitpunkt t = 2s. Der Verlauf von y ist deckungsgleich zu yK1 und wird von diesem überdeckt.

Somit spiegelt diese Simulation die zeitlichen Verhältnisse korrekt wider, wobei die CAN-Wiederholzeit gleich der Schrittweite in den Controllern ist (10 ms).

Sind jedoch die Abtastraten in der Kommunikation zwischen den Teilmodellen unterschiedlich, dann kommt es zu starken Abweichungen im zeitlichen Verhalten der Simulatoren untereinander. Da die Topologie der verteilten Gesamtfahrzeugsimulation ein Stern mit dem Mastermodell im Zentrum ist, bestimmt die im Mastermodell eingestellte Simulationsschrittweite die Synchronisation. Das Problem, welches sich bei verschiedenen Abtastraten in der Kommunikation ergibt, ist ein typisches Simulinkproblem bei mehreren verschiedenen Schrittweiten in einem Modell. In diesem Fall muß mit Anpassungsblöcken, sogenannten Rate-Transition-Blöcken gearbeitet werden, wodurch zeitliche Verschiebungen zwischen den Teilmodellen entstehen.

Beispielhaft wurde das in Abbildung 4.6.3 gezeigte Modell des Masters in Simulink simuliert, wobei das Teilmodell DRV der mit 30 ms simulierte Slave 1 gemäß der Abbildung 4.6.1 ist und das Teilmodell HMI der mit 70 ms simulierte Slave 2.

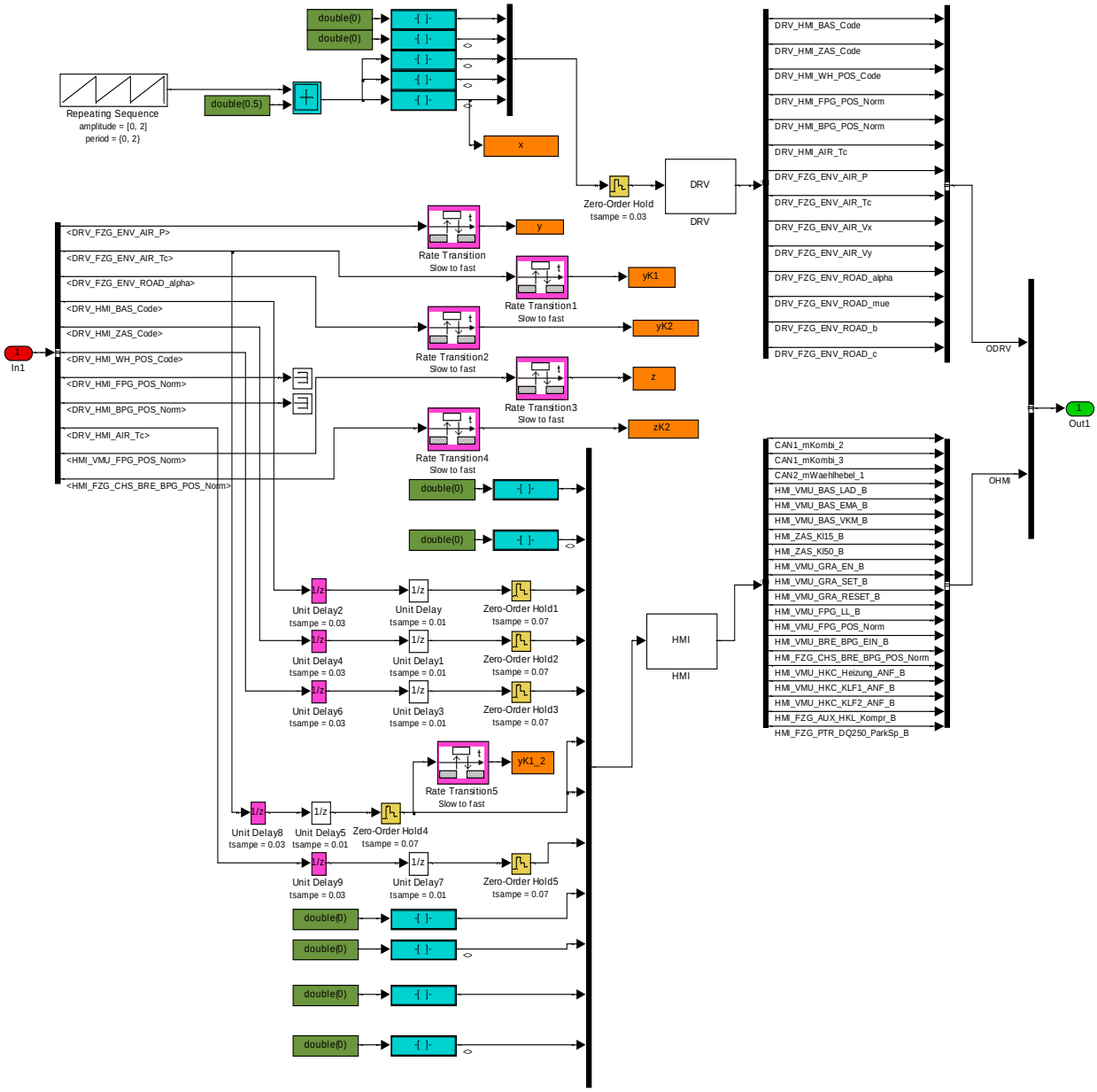


Abbildung 4.6.3: Mastermodell für eine Simulation mit zwei Teilmodellen mit unterschiedlichen Abtastraten in der Kommunikation. Das Mastermodell wird in 10 ms simuliert, Teilmodell DRV in 30 ms und das Teilmodell HMI in 70 ms. Demgemäß ist die Abtastrate der Kommunikation zum DRV 30 ms und zum HMI 70 ms. Der Ausgang dieses Modell ist ausschließlich auf den Eingang zurückgekoppelt.

Wie in der Abbildung 4.6.3 ersichtlich, bereiten die unterschiedlichen Simulationsschrittweiten der Teilsysteme bzw. Abtastraten in der Kommunikation Probleme hinsichtlich der Synchronisation. Simulink generiert Fehler, wenn Systeme unterschiedlicher Simulationsschrittweiten miteinander gekoppelt werden. Aus diesem Grund müssen beim Übergang von schnelleren zu langsameren Systemen Halteglieder oder sogenannte Zero-Order-Hold-Glieder eingefügt werden und beim Übergang von langsameren Systemen zu schnelleren Systemen Rate-Transition-Blöcke, Anpassungsblöcke an die verschiedenen Abtastzeiten. Schwieriger wird es noch, wenn die Schrittweiten der zu verbindenden Blöcke kein ganzzahliges Vielfache voneinander sind. In diesem Fall muß wie bei den Signalen vom DRV zum HMI (Alle Signale mit dem Präfix DRV_HMI_, also z.B. `DRV_HMI_AIR_Tc.`) erst

eine Wandlung in eine Simulationsschrittweite, die gemeinsamer Teiler (optimal größter gemeinsamer Teiler) ist, erfolgen und dann wiederum in die Schrittweite des Zielsystems.

Im Beispiel gemäß der Abbildung 4.6.3 ergibt sich folgender Zusammenhang zwischen den Signalen (s. Abbildung 4.6.4): Das Mastermodell kennt drei Schrittweiten: 10 ms, 30 ms und 70 ms. Diese Abtastzeitpunkte beginnen zu inkrementieren und es bleibt für alle Zeit die Folge 0, 30, 60, 90, … für die 30 ms sowie 0, 70, 140, 210, … für die 70 ms.

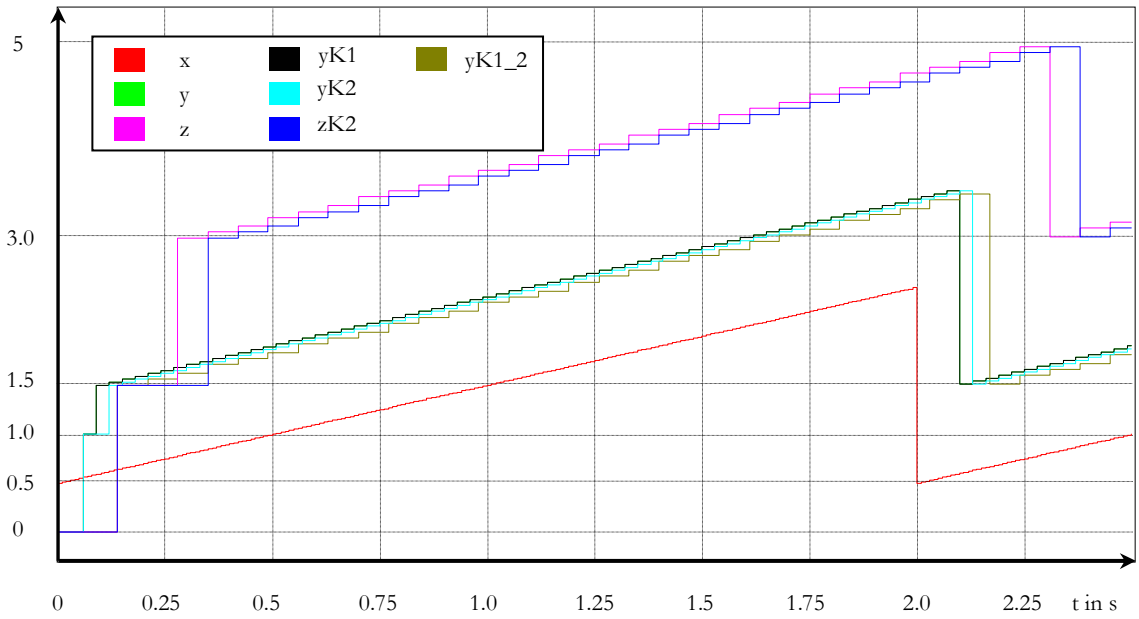


Abbildung 4.6.4: Vermessung einer Simulation mit unterschiedlichen Abtastraten gemäß Abbildung 4.6.3
Der Verlauf von y ist deckungsgleich zu yK1 und wird von diesem überdeckt.

Daneben gibt es noch die 10 ms-Folge. Die jeweiligen Systeme übernehmen die nun anstehenden Werte zu den ihnen eigenen Abtastzeitpunkten. Es ergibt sich ein Jitter, der deterministisch ist aber aufgrund des eben beschriebenen Simulinkproblems unrealistisch. So kommt es nach der Startphase zu der in Tabelle 4.6.2 beschriebenen Verzögerung zwischen den Systemen (s. Abbildung 4.6.4 das Zeitverhalten zum Zeitpunkt t = 2s). Die Auswirkung des Jitterns auf den Signalverlauf ist erkennbar am nicht konstanten Differenzenquotienten der Signale z und zK2. Dennoch ist das zeitliche Verhalten der Simulatoren auch in diesem Beispiel reproduzierbar.

Signal a	Signal b	Verzögerung in Simulationsschrittweiten des Mastermodells (10 ms) zwischen dem Signal b zu dem Signal a	Verzögerung in Simulationsschrittweiten des Mastermodells (10 ms) zwischen dem Signal b zum Ausgangssignal x
x	y	10, veränderlich	10, veränderlich
y	yK1	0, fest	10, veränderlich
yK1	yK2	3, fest	13, veränderlich
yK1	yK1_2	7, fest	17, veränderlich
yK2	z	18, veränderlich	31, veränderlich
z	zK2	7, fest	38, veränderlich

Tabelle 4.6.2: Verzögerung zwischen den einzelnen Signalen gemäß Abbildung 4.6.3
Dargestellt sind die zeitlichen Verhältnisse zum Zeitpunkt t = 2s gemäß der Abbildung 4.6.4. Die Verzögerungen zwischen den Systemen ist nicht wie in Tabelle 4.6.1 konstant, sondern kann veränderlich sein, wenn es sich um Systeme mit unterschiedlichen Abtastraten handelt.

Die Verzögerungen zwischen den Systemen sind ein Vielfaches der Simulationszeit des jeweiligen Systems und haben somit direkt Auswirkung auf das Regelverhalten. Teilsysteme, deren Schrittweiten kein ganzzahliges Vielfaches einander sind, sind zu vermeiden.
Die einzige sinnvolle Alternative zu den eben dargestellten Simulatorkopplungen ist eine einheitliche Abtastrate aller Teilmodelle und die sollte die Simulationsschrittweite des Mastermodells sein. Da in der Gesamtfahrzeugsimulation die Schrittweite des Mastermodells mit 10 ms gleich sämtlicher Abtastraten der Kommunikation mit den Teilmodellen ist, ergibt sich eben genanntes Problem der verschiedenen Abtastraten nicht. Die Schrittweiten der Controller haben nur insofern Einfluß auf das eben genannte Problem, als daß es bezüglich der Kommunikation Jitter geben kann, da sie tatsächlich verteilt, parallel gerechnet werden. (s. z.B. Kapitel 5.2.3 die Verzögerung der simulierten Signale in der Abbildung 5.2.1.)

Jedoch bleibt das Problem der Priorisierung durch Simulink im Mastermodell und der davon abhängigen Abarbeitungsreihenfolge bestehen, die dann einen Einfluß auf die Aktualisierung der Daten der Simulatoren hat. Besser wäre es, wenn es seitens EXITE einen zentralen Synchronisationsmechanismus gäbe, so daß erst nachdem alle Teilnehmerblöcke ihre Daten empfangen haben, die empfangenen Daten zentral weitergeben werden.

Das Verhalten von EXITE wurde mit der Extessy AG kritisch erörtert. Das Ergebnis wird in die zukünftige Version 2.0 von EXITE einfließen, in der das Konzept von EXITE grundlegend überarbeitet und den Bedürfnissen der regelungstechnischen Simulation verteilter Systeme angepaßt wird: So werden Schleifen in den Grafen der Simulatorkopplungstopologien erlaubt. Auch der eben erwähnte Punkt einer zentralen Synchronisation wird durch eine sogenannte „Synchronisationsbarriere“ sichergestellt, indem die Teilmodelle zu einem bestimmten Zeitpunkt gemeinsam ihre empfangenen Daten dem Mastermodell zur Verfügung stellen.

4.7 Systemtheoretische Betrachtung zu Simulatorkopplungen

Grundsätzlich handelt es sich i. allg. bei jedem realen System um ein differential algebraisches System von physikalischen Zusammenhängen in mathematischer Form (DAE; s. Glossar). Durch Umformungen von (Gl. 0.7), beispielsweise durch Substitution und Eliminieren von Variablen, wird normalerweise versucht, das System der Gleichungen zu vereinfachen, um sie in der bekannten Zustandsdarstellung für dynamische, zeitkontinuierliche Systeme darzustellen (Abbildung 4.7.1 bzw. (Gl. 4.7.1)). Ohne Beschränkung der Allgemeinheit läßt sich das System der DGL aus (Gl. 4.7.1) in ein System von n ODEs erster Ordnung darstellen. Der algebraische Ausgangsvektor $\underline{y}$ tritt ausschließlich, linear und explizit in einer algebraischen Gleichung (AGL) auf. Systeme dieser Art sind kausale Systeme mit dem Eingangsvektor $\underline{u}(t)$, dem Ausgangsvektor $\underline{y}(t)$ sowie den inneren Zustandsgrößen $\underline{x}(t)$.

Zustandsgleichung (DGL): $\dot{\underline{x}} = \underline{f}(\underline{x},\underline{u},t)$ (Gl. 4.7.1)

Ausgangsgleichung (AGL): $\underline{y} = \underline{g}(\underline{x},\underline{u},t)$

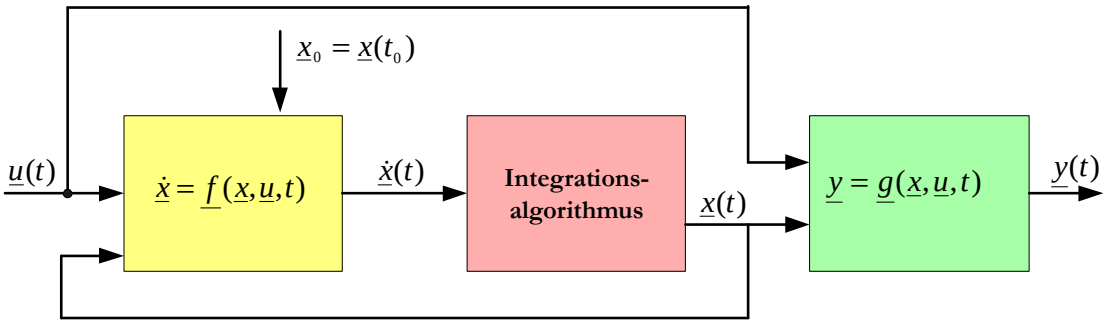


Abbildung 4.7.1: Blockdarstellung eines dynamischen Systems nach (Gl. 4.7.1)
Mit $\underline{x}_0(t)$ werden die Anfangsbedingungen zum Zeitpunkt t = 0 s bezeichnet.

Anhand der Abbildung 4.7.1 werden die drei Schritte bei der Berechnung von Modellen in der Simulation zum Zeitpunkt t_i sichtbar und zwar in folgender Reihenfolge:

1. Schritt: Verknüpfung der Eingangs- mit den Ausgangsgrößen: $\underline{y}(t_i) = \underline{g}(\underline{x}(t_i),\underline{u}(t_i))$
2. Schritt: Bilden der Ableitungen der Zustandsgrößen $\dot{\underline{x}}(t_i) = \underline{f}(\underline{x}(t_i),\underline{u}(t_i))$
3. Schritt: Integrieren des Vektors $\dot{\underline{x}}(t_i)$, um die Zustandsgrößen $\underline{x}(t_{i+h})$ zu erhalten; z. Bsp. mit dem Integrationsverfahren nach Euler ($\underline{x}(t_{i+h}) = \underline{x}(t_i) + h \cdot \dot{\underline{x}}(t_i)$) mit der Schrittweite h (bei äquidistanten Abtastungen bzw. Simulationen mit fixed step-size Solver ist h = T_{ab} ::= die Abtastzeit bzw. die Integrationszeit)

Hierbei wird mit dem 1. Schritt und den Anfangsbedingungen $\underline{x}(t_0)$ begonnen. Der 2. und der 3. Schritt sind unabhängig von der Topologie. Nachteilig auf die Dekomposition und

das Verteilen von Submodellen auf verschiedene Simulatoren ist die Abhängigkeit des 1. Schrittes von der Topologie.
Das eben Gesagte soll an zwei in Serie verschalteten Teilmodellen S1 und S2 erläutert werden:

Die Teilmodelle System S1 und System S2 eines Gesamtsystems S (s. Abbildung 4.7.2) besitzen folgende Systemgleichungen

$$\dot{\underline{x}}_1 = \underline{f}_1(\underline{x}_1, \underline{u}_1, t) \qquad \underline{y}_1 = \underline{g}_1(\underline{x}_1, \underline{u}_1, t) \qquad \text{(Gl. 4.7.2)}$$

$$\dot{\underline{x}}_2 = \underline{f}_2(\underline{x}_2, \underline{u}_2, t) \qquad \underline{y}_2 = \underline{g}_2(\underline{x}_2, \underline{u}_2, t)$$

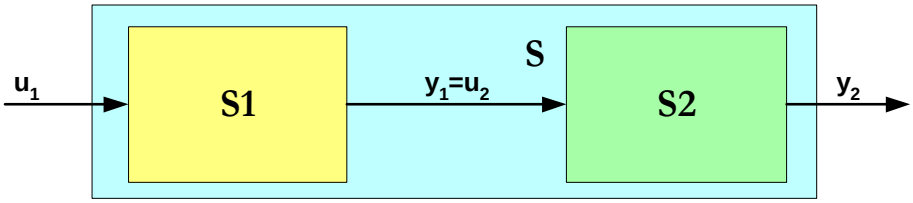


Abbildung 4.7.2: Serienschaltung zweier Teilsysteme

Dann würde die Rechenvorschrift für das Berechnen des Verhaltens des Gesamtsystems zum Zeitpunkt t_i folgendermaßen aussehen:

1. Schritt: Verknüpfung der Eingangs- mit den Ausgangsgrößen:

$$\underline{y}_1(t_i) = \underline{g}_1(\underline{x}_1(t_i), \underline{u}_1(t_i)) \qquad \text{(Gl. 4.7.3)}$$

$$\underline{y}_2(t_i) = \underline{g}_2(\underline{x}_2(t_i), \underline{u}_2(t_i)) \qquad \text{(Gl. 4.7.4)}$$

aber mit $\underline{u}_2(t_i) = \underline{y}_1(t_i)$ (Gl. 4.7.5)

Damit ist (Gl. 4.7.3) von (Gl. 4.7.4) abhängig.

2. Schritt: Bilden der Ableitungen der Zustandsgrößen unabhängig von der Topologie:

$$\dot{\underline{x}}_1(t_i) = \underline{f}_1(\underline{x}_1(t_i), \underline{u}_1(t_i)), \ \dot{\underline{x}}_2(t_i) = \underline{f}_2(\underline{x}_2(t_i), \underline{u}_2(t_i)) \qquad \text{(Gl. 4.7.6)}$$

3. Schritt: Integrieren des Vektors $\dot{\underline{x}}(t_i)$ unabhängig von der Topologie:

z. Bsp. über Euler-Integrationsverfahren wie folgt

$$\underline{x}_1(t_{i+h}) = \underline{x}_1(t_i) + h \cdot \dot{\underline{x}}_1(t_i), \ \underline{x}_2(t_{i+h}) = \underline{x}_2(t_i) + h \cdot \dot{\underline{x}}_2(t_i) \qquad \text{(Gl. 4.7.7)}$$

Die Teilsysteme S1 und S2 sind gemäß (Gl. 4.7.5) über eine Gleichung miteinander verbunden, d.h. die Topologie oder Dekomposition hat einen Einfluß auf die Lösbarkeit des Gesamtsystems. Durch diese algebraische Kopplung, auch algebraische Schleife genannt, würden sprungfähige Teilsysteme, also Systeme wo Eingangsgrößen ohne Verzögerung auf den Ausgang geschaltet werden beispielsweise nur über eine Verstärkung, nicht mehr se-

quentiell rechenbar sein. Um algebraische Schleifen bei aufeinander rückgeführten Systemen zu verhindern, können diese Systeme mit Hilfe nicht sprungfähiger Systeme miteinander gekoppelt werden, z. Bsp. über einen Tiefpaß, eine Totzeit oder ein diskretes Verzögerungsglied. Da dadurch das gesamte Antwortverhalten verändert wird, sollte auf eine realistische Annahme der Zeitkonstanten geachtet werden. Allerdings gibt es in der Realität eine Vielzahl solcher nicht sprungfähiger Systeme. Zu nennen sind Verzögerungen, die durch Laufzeiteffekte auf Signalleitungen und Wandlerschaltungen hervorgerufen werden oder Verzögerungen die durch elektrische oder mechanische Speichersysteme hervorgerufen werden.

Alle in der vorliegenden Gesamtfahrzeugsimulation simulierten Controlleralgorithmen sind zwangsläufig verzögerte Systeme, da die Eingangsgrößen per se um eine Abtastung verzögert werden. Bei den Teilmodellen der Regelstrecke wurde bei der Rückführung von Signalen darauf geachtet, daß ausschließlich integrierte Signale oder durch Energiespeicher hervorgerufene, verzögerte Signale rückgeführt wurden. Zudem besitzt das Tool EXITE die oben beschriebene, fehlerhafte Eigenschaft, selbständig Signale zu verzögern – gerade um algebraische Schleifen zu verhindern. Allerdings kann dies nicht, wie oben beschrieben, Aufgabe eines Kopplungstools sein. Vielmehr hat der Anwender beim Entwerfen der Topologie und der Dekomposition einer verteilten Simulation darauf zu achten, daß keine algebraischen Schleifen entstehen.

Die Topologie der vorliegenden Gesamtfahrzeugsimulation im Sinne der Signaltheorie entspricht der eines vielfach vernetzten, hybriden zeitkontinuierlichen-zeitdiskreten, rückgekoppelten Systems. Zeitdiskrete Regelungs- und Steuerungsalgorithmen greifen an unterschiedlichen Stellen einer verteilt simulierten zeitkontinuierlichen Regelstrecke ein, wie es auch in den Abbildungen 3.4.2 und 3.4.3 angedeutet ist.

Bei verschachtelten Regelalgorithmen unterschiedlicher Teilsysteme, die direkt oder indirekt auf die gleiche Teilregelstrecke wirken, wie beispielsweise im System der Traktionsbatterie/BMS – Elektromotor/Elektromotorsteuerung, sollten diese Regelalgorithmen hinsichtlich des Regelverhaltens, d.h. ihrer Zeitkonstanten ähnlich wie bei den Kaskadenregelungen kaskadiert ausgelegt und miteinander gekoppelt werden, um Schwingungen im Gesamtsystem, hervorgerufen durch Regelschwingungen, zu vermeiden. So wurde darauf geachtet, daß trotz asynchronen Hochlaufs der Steuergeräte in der Realität eine gewisse Kausalität hinsichtlich der zeitlichen Zusammenhänge eingehalten wird: So liefert das BMS Grenzwerte (Strom- und Spannungsgrenzen) für die Elektromotorsteuerung. Diese wiederum regelt mit Hilfe der VMU-Momentenvorgabe den Elektromotor. Die VMU wiederum steuert den Verbrennungsmotor und das Getriebe an, wobei diese Ansteuerungen (zeitlich) plausibel zu Elektromotorsteuerung passen sollten usw. Es ergibt sich also aufgrund der Zeitkonstanten der Regelkreise eine sinnvolle Abarbeitungsreihenfolge der Controller.

Prinzipiell sollen zeitdiskrete Subsysteme realer Systeme, also sämtliche digitalen Regelungs- und Steuerungsalgorithmen und Rechnerfunktionen, zeitdiskret modelliert werden. Der Vorteil besteht in der Rechenzeiteinsparung, da die zeitdiskreten Simulationsmodelle nicht für die numerische Integration des Regelstreckenmodells an allen Zwischenschritten ausgewertet werden müssen. Vielmehr werden die Eingangsgrößen zeitdiskreter Modelle mit einer vorgegebenen Abtastzeit abgetastet, der neue Ausgangswert gebildet mit Hilfe diskreter, schneller Algorithmen (Differenzengleichungen anstatt Differentiation; Summation anstatt numerischer Integration) und dieser mit Hilfe eines Haltegliedes am Ausgang für das nachfolgende, zu simulierende Subsystem konstant gehalten.

Bei der Kopplung zeitdiskreter Modelle mit unterschiedlichen Abtastraten, das sogenannte multi-rate sampling, wäre eine feste diskrete Schrittweite für die Simulation des Regelstrekkenmodells denkbar, die dann dem größten, gemeinsamen Teiler aus gewünschter minimaler Schrittweite der Regelstreckensimulation und aller Schrittweiten der zeitdiskreten Subsysteme entspräche. Im vorliegenden Fall der Hybridantriebstrangsimulation wäre eine diskrete Schrittweite für die Regelstreckensimulation im Bereich von 125 µs und 250 µs denkbar. Um jedoch die bereits erwähnte gewünschte Simulationsgeschwindigkeit zu erhalten, wurden mit Verzicht auf hochtransiente Vorgänge in der Regelstrecke Submodelle ausgegliedert und mit einer festen Schrittweite gerechnet (Vernachlässigung der Momentenripple der Elektromaschine; s. Kapitel 3.2.1) und der interessierende, überwiegend mechanische Teil des Antriebstranges mit variabler Schrittweite. Der Vorteil bei dieser Vorgehensweise besteht darin, daß nun die Zustandsgrößen des zeitkontinuierlichen Systems der Regelstrecke zu den diskreten Zeitpunkten des Datenaustauschs mit den übrigen Teilmodellen unter Anwendung einer Schrittweitensteuerung bei der numerischen Integration ermittelt werden können. Diese Algorithmen sind i. allg. schneller und stabiler als diskrete numerische Integrationsalgorithmen.

5 Validierung der Gesamtfahrzeugsimulation

Daß die in dieser Arbeit vorgestellte Methodik der Gesamtfahrzeugsimulation eine belastbare SIL-Entwicklungsmethodik ist, soll anhand der Validierung der Gesamtfahrzeugsimulation gezeigt werden. Am Beispiel des modellierten Hybrid-Antriebstranges des Volkswagen Bora Hybrid soll die Validierung der Modellierungs- und Simulationsmethodik durchgeführt werden. Dazu werden die Ergebnisse der Simulation direkt mit real, gemessenen Daten verglichen.

Der Inbetriebnahme der hier vorliegenden Gesamtfahrzeugsimulation ging eine Validierung der Regelstrecke und der Funktionsalgorithmen der einzelnen Controller, die nachempfunden wurden und von denen der Code nicht bekannt war, in einer abstrakten Testumgebung voraus. Zu diesen Controllern gehörten das Motorsteuergerät des Verbrennungsmotors, das Getriebesteuergerät des Doppelkupplungsgetriebes, das Elektromotorsteuergerät, das Steuergerät des DC-/DC-Wandlers sowie das Bremsensteuergerät (ABS/ASR). Dazu wurden die einzelnen Controller mit idealisierten Signalen (z. Bsp. Rampe, Sprung o. ä.) stimuliert und mit einer vereinfachten Regelstrecke verbunden. Im Ergebnis lieferten diese Controller plausible Werte und waren in der Lage das jeweilige Aggregat entsprechend anzusteuern.
Der BMS-Controller und die VMU mußten nicht noch einmal validiert werden, da der Code bekannt war und in der vorliegenden Form im Fahrzeug zum Einsatz kam. Es wurden die zum Zeitpunkt der Vermessung des Fahrzeuges identischen Softwarestände der VMU bzw. des BMS in der Simulation verwendet.

Die Frage, wie die zeitliche Abarbeitung der einzelnen Simulatoren in dieser Simulation tatsächlich stattfindet, wird in Kapitel 5.1 geklärt. In Kapitel 5.2 werden anhand eines Fahrzeugstarts durch Einschalten der Zündung die Ergebnisse der Gesamtfahrzeugsimulation mit einer tatsächlichen Messung am Fahrzeug verglichen. Hierbei werden die Daten exemplarisch ausgewählter Controller sowie physikalische Meßgrößen der Regelstrecke qualitativ und quantitativ ausgewertet. In Kapitel 5.3 wird die Gesamtfahrzeugsimulation anhand des NEDC-Zyklus validiert indem die Simulationsergebnisse mit den Meßergebnissen eines Rollenprüfstandversuches verglichen werden. Die verschiedenen Einflüsse auf die Performance der Simulation werden in Kapitel 5.4 untersucht.

5.1 Validierung der zeitlichen Abarbeitung der Simulation

Die zeitliche Untersuchung, wie die Daten der einzelnen Simulatoren weitergereicht werden, ist grundlegend für die Interpretation der Simulationsergebnisse der hier vorgestellten Methodik. Denn die zeitlichen Abläufe innerhalb der Simulation beeinflussen das regelungstechnische Verhalten der einzelnen Controller und der Regelstrecke.
Anhand eines definierten Signalflusses soll die zeitliche Abarbeitung der Simulation im Vergleich mit der Realität validiert werden (Abbildung 5.1.1).

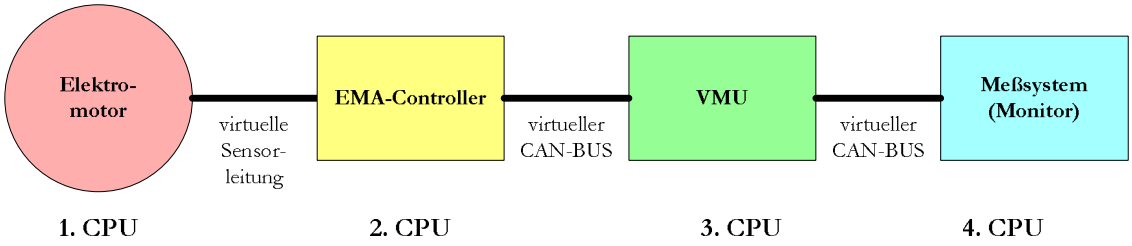


Abbildung 5.1.1: Signalfluß im Modell für die Validierung der zeitlichen Abarbeitung
In jeweils drei verschiedenen physikalischen CPUs werden der Elektromotor einschließlich des EMA-Controllers, die VMU sowie das Meßsystem gerechnet. Verbunden sind die Controller über einen virtuellen CAN-Bus (optisches Netzwerk des Clusterrechners).

Die durch den EMA-Controller gemessene Elektromotoristdrehzahl `EMA_nist` wird der VMU zugeführt. Durch die VMU wird dieses Signal als `VMU_nEMAist` verzögerungsfrei in einer Diagnosebotschaft auf den CAN gelegt.
Wie in der Abbildung 5.1.2 zu sehen ist, ist die zeitliche Verzögerung beider Signale in der Realität wie auch in der Simulation stets konstant und zwar 10 ms. Wie bereits in Kapitel 4.6 gezeigt wurde, gibt die Simulation die zeitlichen Verhältnisse zwischen den einzelnen Kommunikationsteilnehmern richtig wieder. Vor allem auch deshalb, weil die Abtastrate in der Kommunikation bei allen den Teilnehmern in der Abbildung 5.1.1 gleich ist. Aufgrund des in Kapitel 4.6 dargestellten bestehenden Synchronisationsproblems durch die Priorisierung der Simulink-Blöcke im Mastermodell kann es zu geringen Verschiebungen zwischen den Signalen der Teilnehmer kommen.

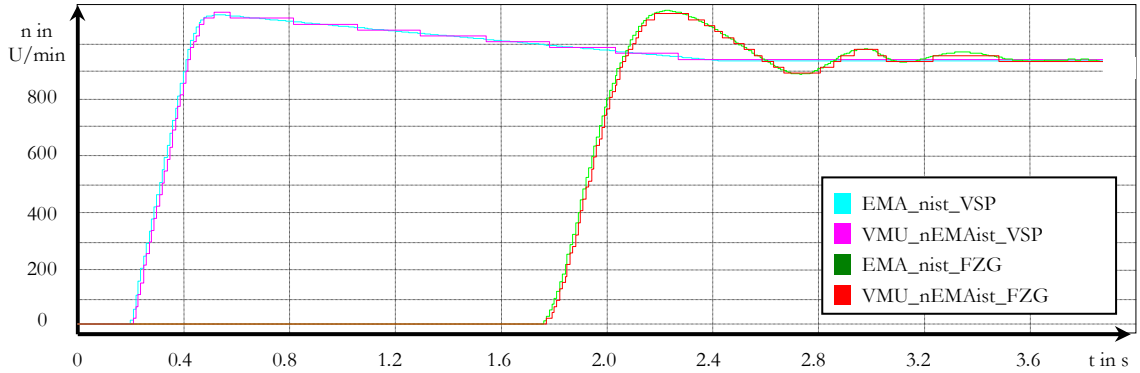


Abbildung 5.1.2: Gegenüberstellung von simulierten Daten (Postfix _VSP) und gemessenen Daten (Postfix _FZG). Simuliert wurde gemäß Simulation (10) in Tabelle 5.4.1.

5.2 Validierung eines Fahrzeugstarts

Dem Test der Einzelsysteme schloß sich die Inbetriebnahme des Simulationsverbundes in Form eines Fahrzeugstarts an. Hierzu wurde mit der Betriebsart „VKM-Betrieb“ begonnen, d.h. der Elektromotor wurde ausschließlich für den Start des Verbrennungsmotors hinzugezogen. Im Einzelnen mußten Zeitkonstanten insbesondere innerhalb der Verbrennungs- und der Elektromotorsteuerung angepaßt werden und das gesamte zeitliche Verhalten anhand eines gemessenen Fahrzeugstarts validiert werden.

Wie bereits erwähnt, wurden 941 der 1059 Signale des Fahrzeug-CAN-Busses simuliert, die alle auf ihre Plausibilität hin untersucht werden müssen und wurden. Am Beispiel ausgewählter Signale soll anhand des Fahrzeugstarts die Übereinstimmung der Simulation mit einer realen Messung gezeigt werden: In den Abbildungen 5.2.1 und 5.2.2 sind sowohl steuergeräteinterne als auch physikalische Größen dargestellt – jeweils als simulierte Größen und im Fahrzeug vermessene Größen. Die Konfiguration der Simulation entsprach der Simulation (8) in Tabelle 5.4.1. Die Namen der gemessenen Fahrzeuggrößen haben den Postfix _FZG und die der simulierten Größen den Postfix _VSP.

In der Abbildung 5.2.1 soll als Beispiel für die Möglichkeiten des virtuellen Prototypings die Interaktion von Steuergeräten untereinander untersucht werden. Es handelt sich hierbei um den Kommunikationsaufbau zwischen der VMU und dem BMS nach dem Einschalten der Zündung. Die Fahrzeugsteuerung in der VMU veranlaßt nach einem vorgegebenen Schema das BMS vom Zustand Parken, d.h. die Schütze der Traktionsbatterie sind geöffnet, in den Zustand Fahren zu wechseln. Im Zustand Fahren sind die Schütze der Traktionsbatterie geschlossen. Da beim Schließen der Schütze die Kondensatoren des Umrichter geladen werden und mithin ein Kurzschlußstrom fließen würde, wird über ein sogenanntes Vorladeschütz ein Widerstand in einen Strompfad geschaltet, um den Einschaltstrom zu begrenzen.

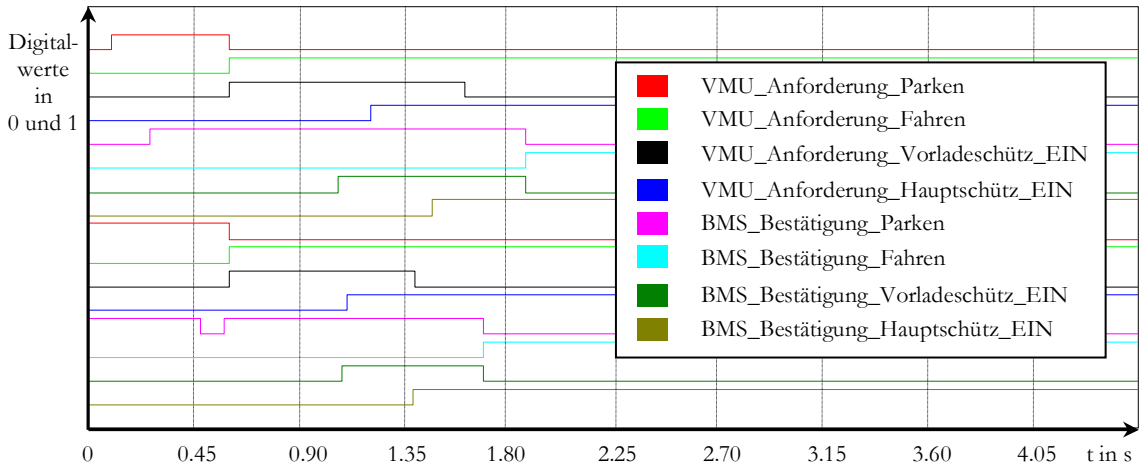


Abbildung 5.2.1: Validierung eines Fahrzeugstarts nach Einschalten der Zündung
Es sind die steuergeräteinternen Flags der VMU und des BMS dargestellt für den Aufbau der Kommunikation untereinander. Jeweils für die Simulation und die gemessenen Daten gilt: sich entsprechende Größen besitzen die gleiche Farbe. In der oberen Hälfte der Grafik sind die simulierten Größen dargestellt. In der unteren Hälfte sind die im Fahrzeug gemessenen Größen dargestellt.

Der Aufbau der Kommunikation zwischen VMU und BMS erfolgt in der Simulation fast identisch zu dem im realen Fahrzeug. Die Meßsysteme sind in beiden Fällen zwischen den

Teilnehmern angeordnet: In der Simulation übernimmt das mit dem Mastermodell gekoppelte Monitormodell die Aufzeichnung und im Fahrzeug eine Meßsoftware. Wie auch im Fahrzeug beträgt die Schrittweite der VMU 10 ms. Das BMS bearbeitet diese Anfrage in der Simulation wie auch im realen Fahrzeug in der 100 ms-Task, d.h. der CAN wird seitens der VMU in 10 ms bedient und seitens des BMS in 100 ms. Der zeitliche Unterschied von Simulation und Messung ist durch die in Kapitel 4.6. beschriebene Synchronisationsproblematik exakt nachvollziehbar. D.h. der zeitliche Unterschied in der Logik ist deterministisch und nachvollziehbar und behindert das virtuelle Prototyping nicht. Wenn beide Steuergeräte die gleiche Abtastrate besitzen würden, dann wäre die Simulation identisch der Messung.

Parallel zum Aufbau der Kommunikation zwischen BMS und VMU, veranlaßt die VMU durch das Regeln der Hydraulikventile die Trennkupplung, die Kupplung zwischen Verbrennungsmotor und Elektromotor, zu schließen (Abbildung 5.2.2). Der durch die VMU gemessene Iststrom des Hydraulikventils `VMU_IistTRK` ist abgesehen von einer Hysterese proportional zum Moment des Ausrückers. In der Abbildung 5.2.2 ist der hysteresebehaftete Schließvorgang der Trennkupplung dargestellt. Da in der Modellierung der Trennkupplung nur die wichtigsten Zeitkonstanten identifiziert wurden, fehlen gegenüber der realen Messung die höheren Frequenzen im Iststrom, was sich dadurch bemerkbar macht, daß der simulierte Iststrom ideal einschwingt. Nach ca. 2 s ist die Trennkupplung geschlossen.

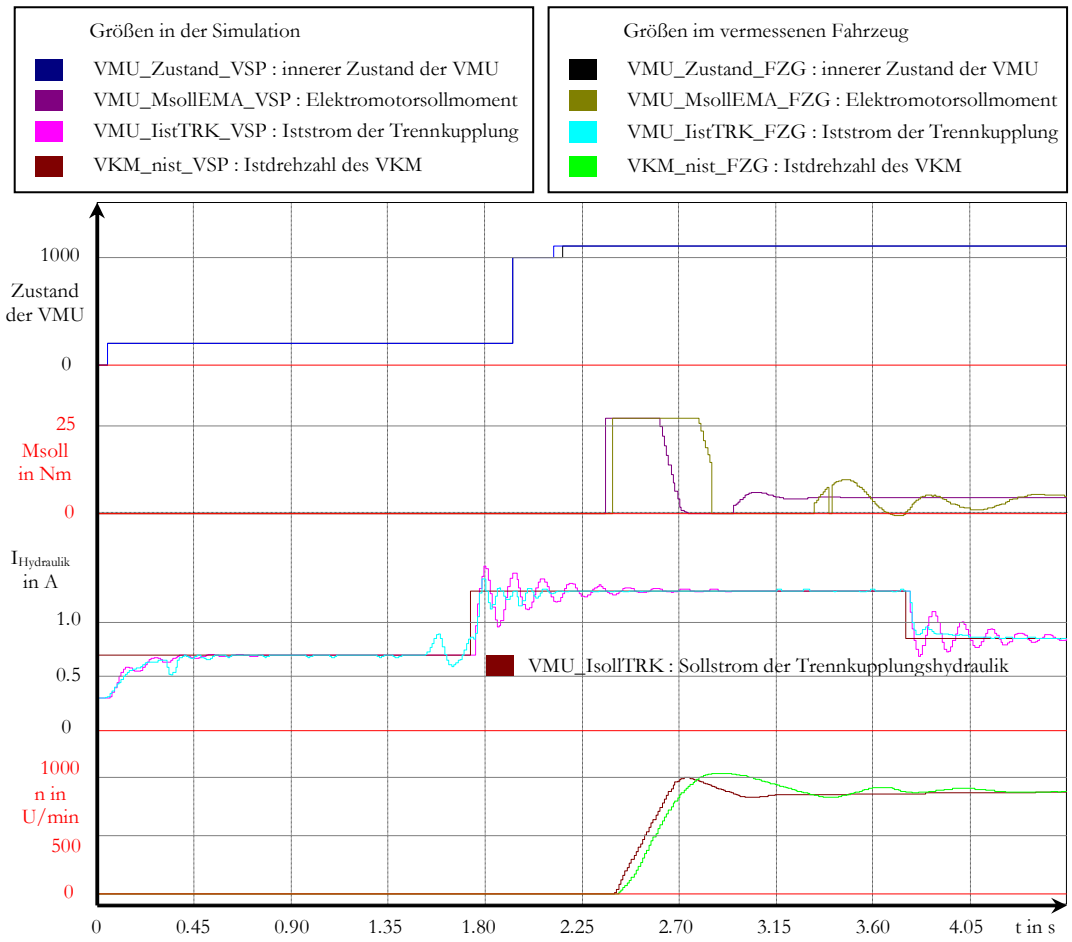


Abbildung 5.2.2: Validierung eines Fahrzeugstarts durch Zündung EIN
Es sind sowohl steuergeräteinterne Größen (`VMU_Zustand`) als auch gemessene Größen (Verbrennungsmotoristdrehzahl, Kupplungshydraulikiststrom) aus Simulation und dem vermessenen Fahrzeug dargestellt.

Der Zustandsautomat in der VMU steuert dabei den gesamten Ablauf nach Einschalten der Zündung. So erfolgt nach Schließen der Trennkupplung die Ansteuerung des Elektromotors durch Vorgabe des Elektromotorsollmomentes `VMU_MsollEMA` durch die VMU, um den Verbrennungsmotor durch Mitreißen zu starten. Nachdem dieser gezündet hat, wird er durch die Leerlaufregelung des Motorsteuergerätes in den Leerlauf gefahren (Istdrehzahlsignal des VKM `VKM_nist`).
Qualitativ und in weiten Teilen auch quantitativ entspricht das in der Abbildung 5.2.2 gezeigte Verhalten der Simulation dem des realen Fahrzeuges. Die Simulation stimmt mit der Fahrzeugmessung sehr gut überein und gibt das Verhalten der Realität sehr gut wieder. Das simulierte Einschwingverhalten der einzelnen Aggregate Verbrennungsmotor, Trennkupplung und Elektromaschine kann durch eine genauere Streckenmodellierung und vor allem durch eine dem realen Fahrzeug identische Ansteuerung der Realität beliebig nahe angepaßt werden – je nach Fokus des Anwenders.

An dieser Stelle soll auf eine weitere Möglichkeit des virtuellen Prototypings in dieser Simulation hingewiesen werden: Es ist möglich, Steuergerätecode Schritt für Schritt auszuführen, zu debuggen, wie es im Englischen heißt, und dabei alle relevanten internen Steuergerätegrößen anzeigen zu lassen bzw. deren Entwicklung schrittweise nachzuvollziehen. Genauer: Wenn die Anwendersoftware eines Controllers zum Beispiel in der Programmiersprache C geschrieben und vorteilhafterweise über compilerunabhängige Datentypen und Funktionen spezifiziert ist, dann kann mit einem Debugger wie zum Beispiel dem des Microsoft Developer Studios für Visual C++ diese Anwendersoftware schrittweise ausgeführt werden. Dabei wird der gesamte Simulationsverbund schrittweise angehalten.
Im Beispiel der BMS wurde diese Möglichkeit untersucht. Diese einzigartige Möglichkeit des Debuggens von Software, die in eine komplexe Regelschleife eingebunden ist, ist eine besondere Eigenschaft des virtuellen Prototypings und nur in einem Simulationsverbund virtuell möglich, in dem die Simulationszeit angehalten werden kann.

5.3 Validierung der Simulation anhand eines NEDC-Rollenprüfstandsversuches

Die Validierung der verteilten Gesamtfahrzeugsimulation erfolgte schrittweise mit der Inbetriebnahme der einzelnen Aggregate und Controller im Gesamtverbund. Dieser sehr zeitintensive Prozeß wurde nicht bis zum Ende durchgeführt. Grund dafür war, wie weiter unten erwähnt, daß die Inbetriebnahme der nachgebildeten Controllerlogik weit mehr Zeit in Anspruch nahm mit all den möglichen Zuständen und Ausnahmen, als die Validierung der Modelle der Aggregate. Dennoch kann an dieser Stelle das Ergebnis der erreichten Validierung dargestellt werden:

Der Referenzkurs, mit dem die in diesem Kapitel beschriebene Validierung durchgeführt werden soll, ist der NEDC. Dazu wurde auf dem Rollenprüfstand durch einen Prüfstandsfahrer mit dem Volkswagen Bora Hybrid ein NEDC-Zyklus abgefahren. Sämtliche Signale des Antriebs-CAN wurden dabei aufgezeichnet.
Die Algorithmen von BMS und VMU in der Simulation sind ***identisch*** den geflashten Algorithmen des Batterie- und Energiemanagementcontrollers im Fahrzeug an dem Tag des Rollenprüfstandsversuches. Das war möglich, da bei diesen beiden Controllern der reale Code in die Simulation eingebunden wurde. Die Modellierung der einzelnen Aggregate und Controller wurde wie im Kapitel 3 erläutert vorgenommen. Die stark gefilterte Istgeschwindigkeit des Fahrzeuges ist die Sollvorgabe im Fahrermodell der Simulation. Grund hierfür sind subjektive Geschwindigkeitsabweichungen, die durch den Prüfstandsfahrer hervorgerufen werden und der Fakt, daß bei freien, nicht vorgegebenen Testfahrten auch nur das tatsächlich gefahrene Istgeschwindigkeitsprofil als Sollgeschwindigkeitsvorgabe für das Fahrermodell in der Simulation herangezogen werden kann. Simuliert wurde die Konfiguration der Simulation Nr. (10) aus der Tabelle 5.4.1. Wie in der Abbildung 5.3.1 zu sehen ist, folgt das virtuelle Fahrzeug in der Simulation der Sollgeschwindigkeit sehr gut.

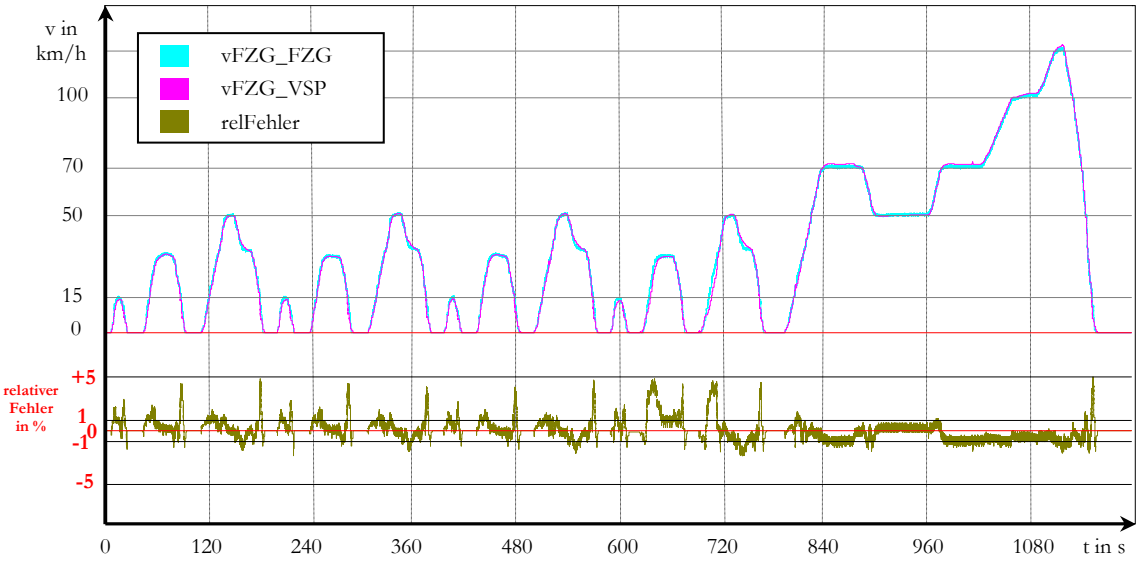


Abbildung 5.3.1: Validierung der Simulation anhand eines Rollprüfstandsversuches (NEDC-Zyklus)
Dargestellt sind die gemessene und die simulierte Fahrzeuggeschwindigkeit (`vFZG_FZG` bzw. `vFZG_VSP`) sowie der relative Fehler `relFehler`.

Verglichen wird die Fahrzeugistgeschwindigkeit des vermessenen Fahrzeuges auf dem Rollenprüfstand `vFZG_FZG` mit der simulierten Fahrzeuggeschwindigkeit des virtuellen Fahrzeuges `vFZG_VSP`.

Dabei wurde der relative Fehler gebildet gemäß der Formel

$$relFehler = \frac{vFZG\_FZG - vFZG\_VSP}{v_{Max}} \cdot 100\% \qquad \text{(Gl. 5.1)}$$

mit v_{Max} = 120 km/h, die Maximalgeschwindigkeit im Zyklus.

Wie man in der Abbildung 5.3.1 sieht, ist der relative Fehler bei dynamischen Geschwindigkeitsänderungen stets kleiner 5 %. und bei konstanten Geschwindigkeiten maximal 1 %. Beide Abweichungen sind akzeptabel und haben ihre Ursache zum großen Teil im Fahrermodell: Denn einerseits wird im Fahrermodell eine Soll-/Istabweichung von ± 2 km/h bzw. ± 1 s zugelassen, wie es im NEDC definiert ist. (± 2 km/h entsprechen ± 1,7 %.) Andererseits treten die Fehler größer 1 % immer bei dynamischen Änderungen auf, also hauptsächlich bei Verzögerungen, die ihre Ursache im Verhalten des Fahrermodells in Verbindung mit dem simulierten ABS-/ASR-System haben, sowie bei Beschleunigungen, die ihre Ursache wiederum im nicht optimal abgestimmten dynamischen Verhalten des Fahrermodells haben. Dabei ist zu beachten, daß sich in dieser virtuellen Antriebstrangsimulation sämtliche Aggregate „drehen" (VKM, GTR, EMA) bzw. elektrische Ströme „fließen" (Traktionsbatterie, DC/DC-Wandler) und von virtuellen Controllern angesteuert bzw. geregelt werden. Vor diesem Hintergrund ist das Ergebnis der Simulation als sehr gut zu bewerten, da es zu keinen Schwingungen im Geschwindigkeitsverlauf kommt (komfortbeeinflussend) und der Fehler akzeptabel klein ist.

Die Abweichungen der Simulation gegenüber der Messung sind weiterhin in der komplexen Interaktion aller Aggregate der Regelstrecke begründet und dabei maßgeblich durch das Schaltverhalten des Direktschaltgetriebes. In der Abbildung 5.3.2 ist am Beispiel des Anfahrens im ECE-Abschnitt des NEDC-Zyklus das reale und das simulierte Schaltverhalten des Direktschaltgetriebes gegenübergestellt (Simulation (9) in Tabelle 5.4.1). Im oberen Teil der Grafik sind die Wellendrehzahlen der Eingangswelle sowie der Wellen 1 bzw. 2 dargestellt, die bei den Gängen 1, 3 und 5 bzw. 2, 4 und 6 das Moment übertragen (s. Kapitel 3.2.1) und im unteren Teil der Grafik die gemessenen und simulierten Kupplungsdrücke der Kupplungen K1 und K2 der Doppelkupplung.
Immer dann, wenn die Drehzahl der Eingangswelle gleich der Drehzahl der Welle 1 bzw. 2 ist, ist die betreffende Kupplung K1 bzw. K2 voll geschlossen und der Schaltvorgang abgeschlossen. In einem solchen Fall wird das Verbrennungsmotormoment über das Getriebe mit der entsprechenden Übersetzung an die Räder weitergegeben.
Wie in der Abbildung 5.3.2, zu sehen ist, sind die Schaltpunkte, d.h. die Drehzahlen, bei denen in den anderen Gang geschaltet wird, sowie die Dauer einer Schaltung unterschiedlich. So dauert die Überschneidung der Kupplungen, also die Zeitdauer wo der Kupplungsdruck der öffnenden Kupplung auf 0 bar reduziert wird und der Kupplungsdruck der schließenden Kupplung auf ca. 2,5 bar erhöht wird, im realen DSG ca. 1…2 s und in der Simulation nur ca. 300 ms. Diese fehlerhafte Applikation der Simulation bewirkt das in der Abbildung zu sehende, schlecht gedämpfte Verhalten des Kupplungsmomentenreglers, und ist am starken Schwingen der Kupplungsdrucks und Wellendrehzahlen zu erkennen.

Da im Simulinkmodell des DSG-Controllers das sogenannte Lastmoment nicht berechnet wird, sind die Schaltkennfelder, ausschließlich wellendrehzahlabhängig. (Das Lastmoment ist das dem Antriebstrang von der Umgebung entgegen gesetzte Moment, hervorgerufen zum Beispiel durch eine Anhängerlast oder Hangabtriebskräfte.) Im realen Fahrzeug jedoch sind diese Schaltkennfelder auch vom Lastmoment abhängig. Aus diesem Unterschied heraus ist begründet, warum zwar die Zeitpunkte der Schaltung in der Simulation und beim realem Rollenprüfstandsversuch in etwa übereinstimmen, aber die Schaltdrehzahlen, bei denen geschaltet wird, unterschiedlich sind.

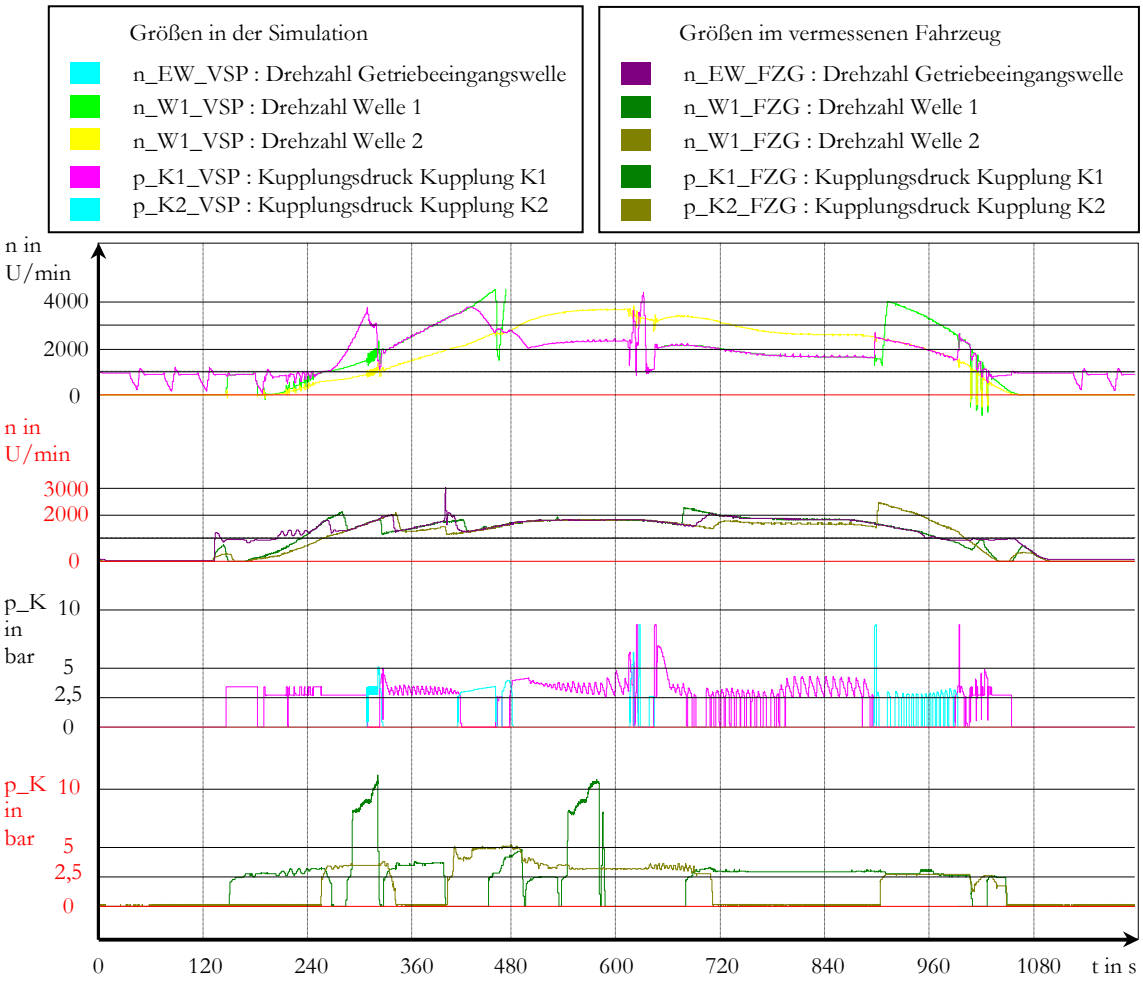


Abbildung 5.3.2: Gegenüberstellung des Schaltverhaltens des realen DSG und des modellierten DSG anhand der Drehzahlen der Wellen und der Kupplungsdrücke. Es sind gemessene, Postfix _FZG, und simulierte Größen, Postfix _VSP, dargestellt.

Besonders anhand des letzten Beispiels kann man erkennen, daß Abstraktionen, auf welcher Ebene auch immer, mit einem Verlust an Genauigkeit in der Simulation einhergehen: Das Weglassen des Einflusses des Lastmomentes führt zu Abweichungen der Simulation gegenüber der Realität. Nicht die Komplexität der Funktionalität sondern der zeitliche Aufwand ihrer Nachbildung in der Simulation war der Grund für diese Vereinfachung.
In dem in der Gesamtfahrzeugsimulation in Matlab/Simulink nachgebildeten Getriebecontroller wurden die Kupplungsmomente bei Übergängen zwischen Zuständen des internen Zustandsautomaten ***geschaltet***. Die Nachbildung der Logik für das weiche Auf- und Absteuern der Kupplungsmomente war zeitlich zu aufwendig. Aus diesem Grund entstehen Schwingungen im Kupplungsdruck, die als Schwingungen bei der Momentenübertragung

bzw. als Drehzahlschwingung auf den Wellen zu erkennen sind (s. Abbildung 5.3.2) und zu Radschwingungen führen. Diese Radschwingungen wiederum bewirken im ungünstigsten Fall Brems- oder ASR-Eingriffe, so daß es rückwirkend wiederum zu einer Rückschaltung des Getriebes in den vorhergehenden Gang kommen kann. So ergab sich aufgrund der abstakten Modellierung des Getriebecontrollers ein stark schwingendes Regelverhalten, welches sich auf den gesamten Vortrieb des Fahrzeugs auswirkte.

Wird, wie in dieser Simulation, der Detaillierungsgrad der Regelstrecke Direktschaltgetriebe sehr hoch gewählt, dann muß auch die Ansteuerung der Regelstrecke entsprechend detailliert modelliert werden. Regelstrecke und Controller bilden eine Einheit hinsichtlich des Detaillierungsgrades. An dieser Stelle soll der Begriff der mechatronischen Einheit geprägt werden:

> *Eine mechatronische Einheit ist dadurch gekennzeichnet, daß sie einerseits einen physikalischen Sachverhalt wiedergibt, also eine Strecke mit ggf. Sensorik und Aktuatorik und zum anderen auch deren Ansteuerung beinhaltet, die analoge oder digitale Schaltungstechnik bis hin zu Mikroprozessortechnik enthalten kann.*
> *Entsprechend des gewählten Abstraktionsgrades der Simulation einer solchen mechatronischen Einheit, müssen Regelstrecke und Ansteuerung auf gleicher Detaillierungsebene modelliert werden.*

Für das DSG in der vorliegenden Simulation folgt daraus, daß aufgrund des stark vereinfachten Controllermodells das DSG hätte abstrakter modelliert werden müssen. Denn der Versuch, einen Controlleralgorithmus für eine relativ detailliert modellierte Regelstrecke nachzubilden, muß am Modellierungsaufwand scheitern. In der Simulation (10) (Tabelle 5.4.1) wurde genau dieser Weg beschritten durch Modellieren des weichen Schaltens mit Hilfe eines idealisierten Schaltgetriebes (Schaltübergänge glatt; s. **(GTR2)** in Kapitel 3.2.1). Aus diesem Grund sind in der Abbildung 5.3.1 keine Sprünge in der simulierten Fahrzeuggeschwindigkeit zu erkennen.
Allerdings, im Fall einer detaillierten Modellierung der Regelstrecke ist es dann zielführend, den realen Funktionscode des Controllers einzubinden. In [Rustemeier et al. 2005] wird eine Methodik beschrieben, wie Funktionscode, der auf realen Steuergeräten fremder Hersteller abgearbeitet wird, in Form einer DLL als sogenannte Black-Box-Modelle in das Gesamtfahrzeugmodell integriert wird.

Die Frage nach dem Detaillierungsgrad der Modellierung einer mechatronischen Einheit sollte vor der eigentlichen Modellierungsarbeit geklärt werden.

Die Validierung soll abgeschlossen werden mit einer Betrachtung zum Verbrauch:
Der Verbrauch ist eine wichtige Ergebnisgröße fast jeder Fahrzeugsimulation, da abgeschätzt werden soll, welche Vorteile die simulierten Konzepte gegenüber dem Bekannten haben. Speziell bei Hybridfahrzeugen ist der Verbrauch eine Kenngröße über die Wirksamkeit der Fahrstrategie und des Hybridkonzeptes.
In einer Botschaft auf dem CAN-Bus legt das Motorsteuergerät den akkumulierten Verbrauch in der Maßeinheit µl ab. Eine Nachfrage bei den Entwicklungsingenieuren für die Motorsteuerungsalgorithmen ergab, daß bei dem in dem Volkswagen Bora Hybrid verbauten Pumpe-Düse-System mit einem maximalen Fehler von ±2 mg pro Einspritzung zu rechnen ist. Bezogen auf das drehzahl- und momentenabhängige Verbrauchskennfeld würden so maximale Fehler im Bereich 2% bis 12 % entstehen. Das Auswerten der entsprechenden Motorbotschaft im auf dem Rollenprüfstand durch das reale Fahrzeug gefahrenen Prüfstandsversuch ergab einen Verbrauch von 3,593 l/100km bei einem anfänglichen Ladezustand der Traktionsbatterie von 70 %. Durch Auswerten der gleichen, simulierten Mo-

torbotschaft ergab sich ein simulierter Verbrauch von 3,598 l/100km, bei annähernd gleichem SOC-Verlauf der Traktionsbatterie.
Dieses überaus positive Ergebnis soll nicht überbewertet werden. Dennoch zeigt es die Tendenz auf, daß die hier vorgestellte Methodik die Realität hinreichend genau widerspiegelt.

5.4 Vermessung der Simulationsgeschwindigkeit

Ein Ziel der hier vorgestellten Gesamtfahrzeugsimulation war zu klären, unter welchen Voraussetzungen die Simulation für den Anwender handhabbar bleibt. Angestrebt wurde eine 20-fache Echtzeit, das heißt, die Simulation soll 20 Mal schneller ablaufen als die in der Simulation gewählte zu simulierende Zeitdauer. Somit würde der NEDC, der als Standardzyklus 20 Minuten dauert, in der Simulation in einer Minute simuliert.
Im Folgenden sollen Einflüsse in der Art und Weise der Modellierung auf die Performance der Simulation aufgezeigt werden:

Der Benchmark wurde unter folgenden Randbedingungen durchgeführt:

- Alle Controller wurden diskret simuliert mit fester Schrittweite entsprechend der Realität (fixed-step Solver).
- Außer dem BMS wurden sämtliche Controller in Matlab/Simulink modelliert.
- Die Abtastzeit der Kommunikation in EXITE ist bei allen Controllern gleich und zwar 10 ms.
- Der Integrationsalgorithmus bei der Modellierung der Regelstrecke in Dymola ist im Allgemeinen `daßl`[1], ein variable-step DAE-Solver, mit einer relativen Toleranz von 10^{-4}. In den Simulationen (4) und (5) wird der Integrationsalgorithmus `lsodar`[1] angewendet.
- Die Kommunikation der Rechner erfolgt über MPI/GM über Myrinet.
- Außer in Simulationen (1), (4) und (5) wurde immer der vollständige NEDC simuliert.

In der Tabelle 5.4.1 sind die Ergebnisse der Laufzeituntersuchungen dargestellt und sollen nachfolgend erläutert und interpretiert werden:

Um die Leistungsfähigkeit des Rechenclusters abzuschätzen, wurden in der Simulation (1) alle Modelle als leere Modelle (sogenannte Dummies) simuliert. Diese Simulation erreichte 67-fache Echtzeit. Ein Simulationsschritt dauert in dieser Simulation ca. 150 µs (Betriebssystemaufruf des Simulators, Kommunikation, usw.).

Von Projektbeginn an bestand Klarheit darin, daß das Modell der physikalischen Regelstrecke Hybridantriebstrang aufgrund seiner Komplexität der zeitliche Engpaß in der Simulation sein wird. Um auch die anderen potentiellen Engpässe abschätzen zu können, sollten die langsamsten Controllermodelle identifiziert werden. Insbesondere aufwendige Matlab/Simulink-Modelle verzögern das Simulieren, wenn diese als Modelle in einer Matlab-Session ausgeführt werden – war die Vermutung. Für Controllermodelle in Matlab/Simulink gibt es die Möglichkeit, diese über den Matlab Real-Time Workshop in ausführbare Dateien (sogenannte executables, RTW-exe oder kurz exe) zu kompilieren. Um die rechenzeitintensivsten Modelle ***nach*** der physikalischen Regelstrecke abzuschätzen, wurden die zwei komplexesten Controllermodelle, der Hybridmanagementcontroller und der Getriebecontroller, kreuzweise jeweils durch ein Dummy und durch die RTW-exe seines tatsächlichen Teilmodells ersetzt. Das Ergebnis waren Simulationen (2) und (3).
In der Simulation (2) verlangsamte der Getriebecontroller bei fehlendem Hybridmanagementcontroller die gesamte Simulation derart, daß selbst, wenn alle anderen Teilmodelle schneller als der Getriebecontroller abgearbeitet würden, nur 5,9-fache Echtzeit möglich wäre.

1 Die Integrationsverfahren daßl und lsodar des Simulationstools Dymola sind in [Elmqvist et al. 2002] erläutert.

In Simulation (3), wo der reale Code der VMU eingebunden wurde, d.h. genau der Softwarestand, mit dem die Rollenprüfstandsmessungen für die Validierung in Kapitel 5.3 durchgeführt wurden, verlangsamte sich die Clustersimulation trotz Kompilieren der VMU in eine RTW-exe auf 8,9-fache Echtzeit, wenn alle anderen Teilmodelle schneller als die VMU simuliert würden. Die Simulation bleibt handhabbar, aber eine 20-fache Echtzeit ist nicht mehr zu erreichen.
An diesem Beispiel erkennt man direkt die Auswirkung der Modellierungstiefe auf die Simulationsperformance: Sehr genau abgebildete komplexe Modelle von Steuergerätealgorithmen in Matlab/Simulink machen jede Simulation unhandlich. Denn es darf nicht vergessen werden, daß eine high-end-Simulationshardware für diesen Benchmark zur Verfügung stand. Wie schon in Kapitel 5.3 erläutert, wäre es bei diesem hohem Detaillierungsgrad des Dymola-Getriebemodells eher zielführend, anstatt des in Simulink nachmodellierten Getriebecontrollers diesen als Originalcode in Form einer vorkompilierten DLL einzubinden. Das hat nicht nur den Vorteil, daß Entwicklungsarbeit eingespart würde, da dieser Controller bereits in C vorliegt und daß das Verhalten dem des echten Controllers entspräche, sondern daß dieser Controller dann auch als DLL um ein Vielfaches schneller abgearbeitet werden würde als ein Modell in Simulink.

Sim. Nr.	Konfiguration	Tsim/ Tmess
(1)	alles Modelle als Dummies –leere Modelle	67,1
(2)	alle ECU als RTW-exe, VMU-ECU als Dummy; FZG: Dymola-Dummymodell	5,9
(3)	alle ECU als RTW-exe, GTR-ECU als Dummy; FZG: Dymola-Dummymodell	8,9
(4)	Wie (10), jedoch Integrationsalgorithmus in Dymola ist `lsodar`	1,5
(5)	Wie (4), jedoch GTR-ECU als Simulink-Modell in Simulationssession	1,56
(6)	FZG-Modell als Dymosim.exe, alle ECU als RTW-exe, EMA-ECU mit 8 kHz shared-memory-Kopplung, GTR-Modell **(GTR 1)**[1], EMA-Modell **(EMA1 b)**[1]	1/390
(7)	wie (6), EMA-ECU mit 4 kHz shared-memory-Kopplung, GTR-Modell **(GTR 1)**[1], EMA-Modell **(EMA 1b)**[1]	1/216
(8)	FZG-Modell aufgespaltet in elektrischen Hochspannungsteil in Matlab/Simulink und Rest in Dymola; alle ECU als RTW-exe, GTR-Modell **(GTR 1)**[1], EMA-Modell **(EMA 2)**[1]	1/14
(9)	Wie (8), jedoch besitzt das Modell des DSG-Getriebes innerhalb des FZG-Modells keine drehzahlabhängigen Reibungen mehr; diese sind jetzt konstant aber ungleich Null, GTR-Modell **(GTR 1)**[1], EMA-Modell **(EMA 2)**[1]	1/2,5
(10)	Wie (8), jedoch DSG-Modell ersetzt durch ein idealisiertes Schaltgetriebe mit konstanten Reibungen, GTR-Modell **(GTR 2)**[1], EMA-Modell **(EMA 2)**[1]	1/1,1

Tabelle 5.4.1: Ergebnis der Laufzeituntersuchung der Simulation
Dargestellt sind für verschiedene Konfigurationen das Verhältnis aus simulierter Zeit Tsim (Angabe der Simulationsdauer im Simulator) und gemessener reiner Simulationszeit Tmess (Die Zeit zum Starten der Modelle und die Zeit des Verbindungsaufbaus ist nicht in diese Auswertung mit eingeflossen.)

Die oben geäußerte Vermutung, daß RTW-kompilierte Matlab/Simulink-Modelle der Controller Performancevorteile gegenüber ausgeführten Simulinkmodellen haben, konnte nicht bestätigt werden: In den Simulationen (4) und (5) sind zwei Simulationen gegenübergestellt, wo einmal die GTR-ECU als RTW-exe und einmal als Simulinkmodell simuliert wurde. Es ist nur ein kleiner, irrelevanter Unterschied in der Performance zu erkennen.

1 Die Modellierung der Modelle **(EMA 1b)**, **(EMA 2)**, **(GTR 1)** und **(GTR 2)** sind in Kapitel 3.2.1 beschrieben

Allerdings dauert der Aufruf des Simulinkmodells beim Start der Simulation entsprechend der Kompilezeit ein Vielfaches länger als der Aufruf des RTW-kompilierten Modells und stoppt so den gesamten Simulationsverbund. (Diese Zeit des Verbindungsaufbaus einschließlich des Interpretierens und Startens des Simulinkmodells ist in der Tabelle 5.4.1 nicht mit eingeflossen.)

Beide Simulationen sind schneller als Echtzeit, allerdings nicht stabil und brechen mitten in der Simulation ab. Grund dafür ist der Solver `lsodar`, der nur ODE-geeignet ist.

Nicht in Tabelle 5.4.1 aufgelistet ist der Versuch, das virtuelle Fahrzeug mit allen seinen Teilmodellen und detaillierten Sensor-/Aktorschaltungen in Dymola zu simulieren, da diese Simulationen zu unhandlich langsam wurden. So wurde der Datenaustausch zwischen EMA-ECU und Regelstrecke sowie das digitale PWM-Signal zur Ansteuerung des Umrichters mit 6 kHz Schaltfrequenz simuliert. Dies hat zur Folge, daß alle 1/6000 s (Zeitintervall in der Simulation) in Dymola ein Ereignis, ein sogenanntes time-event, ausgelöst wird, um die Zustandsgrößen zu diesem Zeitpunkt zu berechnen. Aus diesem Grund wird der Solver durch eine Schrittweitensteuerung veranlaßt, diesen Simulationszeitpunkt genau zu treffen, was eine drastische Verschlechterung der Simulationsperformance zur Folge hat. Diese sehr detaillierte Modellierung des Antriebstranges einschließlich des Umrichters mit B6-Brücke und Ansteuerung des Umrichters mit im Unterschwingungsverfahren generierten Ansteuersignale (s. **(EMA 1a)** in Kapitel 3.2.1) war nicht mehr handhabbar simulierbar. Die Simulation einer Sekunde „Fahrt“ dauerte weit mehr als 5 Minuten. Im Ergebnis allerdings konnten die Momente der Elektromaschine untersucht werden und somit die Auswirkungen des Schaltverhaltens des Umrichters auf die Momentenoberwellencharakteristik.
Generell stellt sich allerdings die Frage, ob wirklich jede Sensor- oder Aktorschaltung modelliert werden muß oder ob die Fehler, die durch eine Abstraktion entstehen nicht vernachlässigbar sind.
Als Erkenntnis aus dieser Modellierung kann abgeleitet werden, daß vor dem Beginn einer Modellierung generell überlegt werden muß, inwieweit detailliert modellierte Sensor- und Aktorschaltungen überhaupt einen Beitrag zum Simulationsergebnis bzw. zum Erkenntnisgewinn aus einer Simulation bringen.

Das durch das Unterschwingungsverfahren generierte Puls-Weitenverhältnis, welches die Information über die Ansteuerspannung der Ventile des Umrichters enthält, wurde in den folgenden Simulationen ersetzt durch eine dem PWM-Verhältnis proportionale Ansteuerspannung. Genauso wurde bei dem PWM-Signal des BMS als Ansteuersignal des PWM-geregelten Batterielüfters vorgegangen: Wenn die Fehler im Simulationsergebnis, die durch diese Vereinfachung gemacht werden, vernachlässigbar sind oder wenn diese Schaltung nicht spezielles Interesse der Simulation ist, dann sollte auf das Generieren des Puls-Weitenverhältnis verzichtet werden.
Diese Aussage trifft auf alle Sensor- und Aktorschaltungen zu, wo in irgendeiner Weise virtuelle Ansteuersignale physikalischer Größen über eine elektronische Schaltung wieder in physikalische Größen transformiert werden und umgekehrt. Statt dessen kann auf einer relativ hohen Abstraktionsebene in der Softwarearchitektur ein Schnitt gemacht und es wird die Funktionssoftware mit physikalischen Größen gerechnet, die dann direkt auf die Regelstrecke wirken. Ausnahmen bilden Flags der Controller, die Schalter in der Regelstrecke ansteuern.
Werden Controlleralgorithmen, wie das BMS, in Integerarithmetik abgearbeitet, dann muß eingangsseitig die Sensorkennlinie mit der Quantisierung des A/D-Wandlers nachgebildet werden.

Diese Methodik wurde in den nachfolgenden Modellierungen angewendet: So wurden sämtliche PWM-Signale ab einschließlich Simulation (4) als kontinuierliche Signale mit der Größe ihres Wertes proportional zur Pulsweite modelliert.
In der Simulation (6) betrug die Abtastrate der Kommunikation zwischen EMA-ECU und Regelstrecke und somit die Schrittweite von time-event zu time-event im physikalischen Modell 125 µs, obwohl diese Kommunikation nicht über das Netzwerk realisiert wurde sondern über eine shared-memory-Kopplung zweier Prozessoren eines Doppelprozessorsystems. Diese Kommunikationsart ist die schnellste zwischen zwei Recheneinheiten. Mit dieser nun hinsichtlich des Umrichters vereinfachten Simulation des Systems Elektromotor ließ sich eine Performance von ca. 390-fach langsamer als Echtzeit erreichen. Auch eine Halbierung der Abtastzeit der Kommunikation zwischen EMA-ECU und Regelstrecke brachte nicht den Sprung in der Performance (s. Simulation (7) in Tabelle 5.4.1). Der Grund dieser schlechten Performance liegt in der Frequenz der Kommunikation zwischen EMA-ECU und Regelstrecke: Bei jedem Datenaustausch zwischen EMA-ECU und Regelstrecke wird im Dymola-Solver ein time-event ausgelöst, so daß der Integrator gezwungen wird, diesen Simulationszeitpunkt genau zu treffen. Die Art und Weise, wie diese time-events vom Solver behandelt werden, hat wesentlichen Einfluß auf die Performance.

Will man die Abarbeitungsgeschwindigkeit der Simulation steigern, so muß versucht werden, mit hinnehmbaren Genauigkeitsverlusten die Simulation zu parallelisieren. Besonders rechenzeitintensive Modelle, wie zum Beispiel das Modell des Elektromotors oder das Modell des Getriebes sollten parallel zu dem restlichen Modell der physikalischen Regelstrecke auf getrennten Prozessoren abgearbeitet werden. Das Elektromotormodell mit seiner Ansteuerung in Schaltfrequenz durch das vereinfachte Modell des Umrichters wurde vom physikalischen Modell getrennt und zusammen mit dem Teil des elektrischen Systems, welches auf Traktionspotential liegt einschließlich des Traktionsbatteriesystems sowie der EMA-ECU mit Vektorregelung gemäß [Leonhard 2000] auf einem zweiten physikalischen Prozessor eines Doppelprozessorensystems abgearbeitet. Diese Konfiguration wurde in **(EMA 2)** in Kapitel 3.2.1 bereits erläutert. Die Zeit zum Weiterschalten des Feldvektors beträgt je nach Schaltfrequenz sinnvollerweise zwischen 100 µs bis 250 µs, damit keine nennenswerten Momentenripple entstehen. Dies ist auch die Dauer der Task, die das dem Sollstrangstrom proportionale Ansteuersignal für das vereinfachte Modell der Leistungselektronik berechnet. Alle 250 µs müßte der Integrator des Dymola-Solvers zurückgesetzt werden, wenn der Umrichter und der Elektromotor Teil des Dymolamodells geblieben wäre. Somit erfolgte ab Simulation (8) die Kopplung des Elektromotormodells in Matlab/Simulink mit dem restlichen physikalischen Modell in Dymola über Momente und Drehzahlen im 10 ms Takt und enthält nicht mehr die hochdynamischen Anteile, die von der Modellierung des Schaltens des Umrichters herrühren. Mit dieser Konfiguration wurde eine Simulationsgeschwindigkeit von ca. 14-fach langsamer als Echtzeit erreicht.

Nach Rücksprache mit der Firma Dynasim bezüglich weiteren Potentials zur Verbesserung der Simulationsperformance, wurden in der Simulation (9) die in Simulation (8) drehzahlabhängigen Reibungskennlinien der einzelnen Übersetzungsstufen des DSG durch konstante Reibungen ungleich Null ersetzt. Denn ein weiteres Problem des Solvers sind physikalische events, sogenannte state-events, die durch Steifigkeiten im Antriebstrang hervorgerufen werden. In diesem Fall wurde die Performance weiter verbessert zu 0,4-facher Echtzeit. Wenn ganz auf Reibung im Getriebe verzichtet wird, erreicht die Simulation beinahe Echtzeit (Simulation (10)).

Der Fokus, der in diesem Kapitel durchgeführten Simulationen, galt weniger dem Vergleich der Genauigkeiten der Simulation als der Handhabbarkeit. Die Genauigkeit ist in jedem Fall bezüglich des Gegenstandes oder des Ziels der Simulation abzuschätzen. Es wurde

gezeigt, daß ausgehend von sehr komplexen Simulationen durch Anwenden von Parallelisieren, Vereinfachen von Modellen (Vernachlässigen bzw. Vereinfachung der Reibung etc.) und durch geeignetes Abstrahieren (geeignetes Modellieren von Sensor- und Aktorschaltungen) die Performance wesentlich beeinflußt werden kann. In den meisten Fällen liegt das Problem in der Komplexität der Modelle und dem Vermögen des verwendeten Solvers. Ersteres gilt für sämtliche Simulatoren; letzteres ist stark abhängig vom verwendeten Simulator. Die verschiedenen Simulationstools für Regelstreckensimulationen sind hinsichtlich der Handhabbarkeit in der Modellierung alle ähnlich gut. Sie unterscheiden sich nur hinsichtlich der Effizienz des Solvers (Methoden, Lösungsverfahren). Insoweit muß mit dem Toolhersteller eng zusammengearbeitet werden, wenn es um eine weitere Verbesserung der Simulationsperformance geht.

6 Zusammenfassung und Schlußfolgerungen

Die vorliegende Arbeit zeigt, daß virtuelles Prototyping in der Automobilindustrie die Entwicklung von Steuergerätealgorithmen sowie das Bewerten ausgelegter Aggregate ermöglicht und zu verwertbaren Ergebnissen führt. Die Realität läßt sich wie gezeigt abhängig von der gewählten Modellierungstiefe sehr genau widerspiegeln. Das virtuelle Prototyping ermöglicht ein teilweise zeitgleiches Entwickeln von Teilaspekten eines Entwicklungsgegenstandes im Gegensatz zu der gegenwärtigen sequentiellen Vorgehensweise von Entwicklung, Test und Applikation.
Am Beispiel der Modellierung eines Hybridantriebstranges eines Kraftfahrzeuges wurde die gesamte Steuerungslogik eines Hybridfahrzeugantriebstranges virtuell abgebildet und gezeigt, daß diese Methodik für die Software-in-the-loop-Entwicklung von Steuergerätecodealgorithmen geeignet ist. Somit können komplexe, vernetzte und/oder überlagerte Regelungssysteme entwickelt und robust ausgelegt werden.

Die Ergebnisse lassen sich in den folgenden Punkten zusammenfassen:

(1) Die Wahl des Detaillierungsgrades von Modellen und Teilmodellen muß in Bezug auf das Ziel der Simulation genau begründet und festgeschrieben werden.
Zu detailgetreue Modelle bringen oft keine Verbesserungen in den Simulationsergebnisse sondern statt dessen nur Probleme in der Handhabbarkeit und Rechenbarkeit der Simulation mit sich. Abhängig vom zu untersuchenden Teilaspekt sollte jedes Teilmodell hinsichtlich seines Detaillierungsgrades und der Abhängigkeit zu den anderen Teilmodellen modelliert werden. Diese grundsätzlichen Überlegungen müssen ***vor*** dem Beginn der Modellierung gemacht werden.

(2) Bei der Modellierung einer mechatronischen Einheit, d.h. der physikalischen Regelstrecke Aggregat und der dazugehörigen Steuerungslogik, darf die Abstraktionsebene, also der Detaillierungsgrad nicht geändert werden.
Physikalische Aggregate, wie zum Beispiel Motor oder Getriebe, werden wie eben beschrieben mit einem bestimmten Ziel der Simulation durch den gewählten Detaillierungsgrad auf einer bestimmten Abstraktionsebene modelliert.
Wenn zu einem solchen Aggregat eine elektronische Steuerung in Form eines Steuergerätes gehört, so muß der Detaillierungsgrad dieser Steuerung dem des Aggregates entsprechen. Es hat keinen Sinn, einen Teil der Regelstrecke aufwendig zu modellieren, wenn er mangels Detailtreue des Steuergerätealgorithmus' nicht adäquat angesteuert werden kann.
Somit bilden das Modell des Aggregates und der dazugehörige Steuerungsalgorithmus eine (mechatronische) Einheit.

(3) Es ist durchaus möglich, bei verschiedenen mechatronischen Einheiten einer Simulation verschiedene Abstraktionsebenen anzuwenden. Allerdings muß bei der Kopplung dieser Systeme der dabei entstehende systemtheoretische bzw. regelungstechnische Fehler bewertet werden.

(4) Die genaue Nachbildung von Steuergerätecode führt nicht oder nur mit sehr hohem Aufwand zum Ziel.

Je genauer die Detaillierung der Regelstrecke gewählt wird, desto ernsthafter muß darüber nachgedacht werden, Steuerungsalgorithmen in Form von realem Steuergerätecode einzubinden. Der Aufwand einer genauen Nachbildung von Steuergerätealgorithmen steht in keinem Verhältnis zu dem Ergebnis, welches man in der Simulation dadurch erreichen kann. Das Nachführen von Änderungen gemäß realer Algorithmen und Sonderfälle sind eher potentielle Fehlerquellen in der Simulation.

(5) Sensor- und Aktorschaltungen sind zu abstrahieren.

Falls Sensor- und Aktorschaltungen oder nachgeschaltete Modelle nicht Focus und Gegenstand der Untersuchung einer Simulation sind, bringen detaillierte Sensor- und Aktorschaltungen keinen wesentlichen Beitrag zur Genauigkeit der Simulation. Sensor- und Aktorschaltungen sind Teil der Modellierung der Regelstrecke und sollten wie auch die dazugehörigen Digital-/Analog- oder Analog-/Digitalwandler abstrahiert werden. So sollte als Schnittstelle der Controller zu dem Modell der physikalischen Regelstrecke mit physikalischen Signalen in SI-Einheiten gearbeitet werden und PWM-Ansteuerungen durch äquivalente kontinuierliche physikalische Größen substituiert werden.

Das virtuelle Prototyping kann in der Produktentwicklungsphase entscheidende Wettbewerbsvorteile mit sich bringen unter anderem durch:

- Senken der Entwicklungskosten,
- Verkürzung der Entwicklungszeit und
- Senken der Kosten nach dem Start der Produktion (SOP).

Des weiteren können durch die reproduzierbaren Simulationen des virtuellen Prototypings Regler und Regelstrecken robust ausgelegt werden, unter Bedingungen, wie sie mit vertretbarem Aufwand und Kosten nur in der Simulation gegeben sind. Parametervariationen und Varianzanalysen können massiv parallel gerechnet werden.

Die in dieser Arbeit vorgestellte Methodik des virtuellen Prototypings ist nicht auf die Simulation von Antriebsträngen von Kraftfahrzeugen beschränkt. Vielmehr kann diese Methode angewendet werden bei:

- der Modellierung jeglicher physikalischer Systeme sowie
- dem Frontloading von mikrocontrollergesteuerten Systemen jeder Art.

Das virtuelle Prototyping bietet die Möglichkeit eines Software-Integrationstestes sowie eines Systemtests vor dem Prüftstandsversuch und dem Fahrzeugaufbau. Eine validierte, echtzeitfähige SIL-Simulation kann durch das Herausführen der entsprechenden Steuergeräteschnittstellensignale in einen HIL-Simulator überführt werden.

Somit ermöglicht das virtuelle Prototyping:

- in einer sehr frühen Projektphase, virtuell und ***gleichzeitig*** mit der Hardware-Konzeptauslegung die Steuergerätefunktionssoftware zu spezifizieren, zu entwikkeln (auch zu debuggen !) und zu testen (Schlagwort „digitales Lastenheft“),
- den Verzicht auf einen Hardwareverbau realer Steuergeräte in einer frühen Projektphase (Stichwort Breadboard) sowie

- mit vertretbarem Aufwand die komfortable Simulation physikalischer Systeme mit großen Zeitkonstanten und hoher Parametervielfalt und damit den Verzicht auf ***aufwendige*** Laboruntersuchungen.

Beim Aufbau eines Simulators oder Simulationsverbundes mit dem Ziel des virtuellen Prototypings muß insbesondere darauf geachtet werden,

- daß die Simulation handhabbar ist, d.h. schneller als Echtzeit,
- daß die Simulation regelungstechnisch der Realität entspricht und
- daß die Simulation ausreichend genau ist, d.h. die Realität hinreichend genau widerspiegelt.

Diese Aspekte wurden in der vorliegenden Arbeit untersucht und es wurde eine Methodik vorgestellt, die das virtuelle Prototyping ermöglicht.

Durch die objektorientierte Modellierung in Modelica wird ein hoher Grad an Flexibilität hinsichtlich des Konfigurierens neuer Systeme, aufbauend auf vorhandenen Modellen, sowie an Übersichtlichkeit, d.h. Modellierungsqualität erreicht. Dies wird zudem seitens des Tools Dymola durch eine Parametrierungsoberfläche auf der Ebene der grafischen Modellierung unterstützt. Da in Zukunft an einem integriertem Konfigurations- und Variantenmanagement für das Verwalten von Teil- und Gesamtmodellen gearbeitet wird und ab Version der 6.0, welche im Laufe des Jahres 2006 verfügbar sein wird, eine Monte-Carlo-Analyse integriert sein wird, steht dieses Tool den bekannten leistungsfähigen, objektorientierten Multidomän-Simulationstools wie ADAMS oder SABER in nichts nach.

7 Ausblick

Die prinzipielle Frage, ob mit Hilfe einer dynamischen Gesamtfahrzeugsimulation Frontloading möglich ist, wurde in dieser Arbeit beantwortet. Wie am Beispiel der folgenden beiden Punkte gezeigt wird, bleibt genügend Spielraum für weiterführende Arbeiten, um die vorliegende Gesamtfahrzeugsimulation noch leistungsstärker und komfortabler zu gestalten.

1. Performancesteigerung der Clustersimulation

Trotz der wie Kapitel 5.4 erwähnten 0,9-fachen Echtzeit der Simulation eines NEDC-Zyklus' würde ein Simulationsdurchlauf, also eine Parametervariation ca. 18 Minute dauern. (Nicht eingerechnet sind Neukonfiguration und Start der nächsten Simulation.) Dies ist bei mehr als vier Variationen nicht mehr handhabbar, da in diesem Fall frühestens erst nach einer Stunde die Ergebnisse vorliegen würden. Es sollen fünf Möglichkeiten zur Performancesteigerung genannt werden:

(1) Schon in [Soejima et al. 2002] wurde auf die Problematik der Wahl des richtigen Solvers auf die Simulierbarkeit eines Modells aufmerksam gemacht. Da das derzeitige Problem die Performance der Regelstreckensimulation ist, müssen die Möglichkeiten von Dymola genauer untersucht werden: Neben der Auswahl verschiedener Integrationsalgorithmen kann auch die Inlineintegration zielführend sein, eine kombinierte symbolische und numerische Herangehensweise für die Lösung differential algebraischer Gleichungssysteme. Wichtigster Punkt jedoch ist das Beschleunigen der Simulation durch spezielle Solver, die unterscheiden zwischen time-events und state-events: Da der Datenaustausch über EXITE und über die erwähnte shared-memory-Kopplung zu festen und ***bekannten*** Zeiten erfolgt, muß der Solver gezwungen werden, hinsichtlich seiner Schrittweitensteuerung diese Abtastzeiten der Kommunikation genau zu treffen. Zusätzliche oder sogar Rückwärtsschritte wären dann nicht mehr notwendig. Wie schon am Ende des Kapitels 5.4 erwähnt, sollte bei all diesen Überlegungen eng mit dem Hersteller des Simulationstools zusammengearbeitet werden.
Eine Möglichkeit, die nachteiligen Auswirkungen der time-events auf den Solver zu umgehen, ist das Festlegen einer maximal erlaubten Schrittweite für die sogenannte dense output Methoden spezieller Solver: Diese Methoden behandeln Punkte in der Kommunikation, die time-events auslösen würden, unterschiedlich als dies üblicherweise Solver tun. Diese Zeitpunkte müssen bei speziellen Solvern nicht mehr genau getroffen werden, denn die Lösung zum entsprechenden Zeitpunkt in der Kommunikation wird interpoliert.
Weiterhin kann bei einem gewählten Solvers mit der Ordnung der Integrationsmethode experimentiert werden, um die Simulationsgeschwindigkeit zu erhöhen. Sie sollte auf einen festen Wert gesetzt werden, da bei höherer Ordnung der Integrationsmethode die Integrationsschrittweite bei gleichem maximalen lokalen Fehler höher gewählt werden kann und somit die Simulation effektiver ausgeführt wird.

(2) Die Steifigkeiten des Systems müssen unter Berücksichtigung des sinnvoll gewählten Detaillierungsgrades des Teilmodells untersucht und möglichst eliminiert werden.

(3) Nach weiteren Teilsystemen in der physikalischen Regelstrecke sollte gesucht werden, um diese sinnvoll auszulagern, wie es am Beispiel des Traktionsspannungssystems in Kapitel 5.4 gezeigt wurde. Dabei muß jedoch auf eine sinnvolle physikalische Kopplung der

Systeme geachtet werden, um zeitlich transiente Zusammenhänge im Rahmen der geforderten Genauigkeit plausibel zu simulieren.

(4) Im Falle zu langsam simulierter Controller können diese ohne weiteres in mehrere parallel abzuarbeitende Module zerlegt werden, wenn es die Softwarestruktur zuläßt.

(5) Da, wie erwartet, die Simulation der physikalischen Regelstrecke Fahrzeug-Straße aufgrund der relativ aufwendigen Modellierung die Performance der Gesamtfahrzeugsimulation bestimmt, liegt es nahe, über eine mögliche Parallelisierung der Simulation nachzudenken. Vorstellbar ist ein Parallelisieren per Hand oder das automatische Parallelisieren des Simulationscodes. Ersteres scheidet aus, da die Simulation handhabbar bleiben soll und das physikalische Modell nur ***ein*** Teil der Gesamtfahrzeugsimulation ist. Die automatische Parallelisierung ist ein realistischer Weg, um die Performance der Gesamtsimulation wesentlich zu steigern und wurde in [Aronsson 2002] am Beispiel der Modellierungssprache Modelica untersucht. Es wurde ein Tool entwickelt, welches eine effiziente, parallele Version des Simulationscodes durch Ableiten eines Abhängigkeitsgrafen (Task Graph) vom Simulationscode und Anwenden effizienter Scheduling- und Clustering-Algorithmen erstellt.

2. Testautomatisierung

Um eine massive Parametervariation einschließlich der Datenaufzeichnung automatisch durchführen zu können, kann diese automatisiert werden. Mögliche Ansätze wären zum Beispiel entweder eine über Scripte automatisierte Parametervariation durch vorher definierte Parametersätze. Auch ist eine Monte-Carlo-Analyse denkbar, wo sich der Parametersatz innerhalb festgelegter Grenzen zufällig ändert.

8 Literaturverzeichnis

[Aronsson 2002] P. Aronsson: Automatic Parallelization of Simulation Code from Equation Based Simulation Languages, Licentiate Thesis, Linköping University, Sweden, 2002

[Autosar 2004] Autosar Arbeitskreis, http://www.autosar.org

[Barth et al. 2002] Thomas Barth und Manfred Grauer: Grid Computing-Ansätze für verteiltes virtuelles Prototyping; Universität Siegen, erschienen in: D. Schoder, K. Fischbach, R. Teichmann (Hrsg.), Peer-to-Peer (P2P), Springer Verlag, August 2002

[Beuschel 2000] M.Beuschel: A Uniform Approach for Modeling Electrical Machines, Modelica Workshop 2000 Proceedings, pp. 101-108

[Biermann et al. 2004] J.-W. Biermann, C. Bunz, M. Crampen, S. Köhle, D. Mesiti: Drei OEM, ein gemeinsames Antriebskonzept – Drei neue Hybridfahrzeuge, entwickelt im EU-Projekt SUVA, 13. Aachener Kolloquium Fahrzeug- und Motorentechnik, Aachen, 2004

[Bikker et al. 2002] G. Bikker, M. Schroeder: Methodische Anforderungsanalyse und automatisierter Entwurf sicherheitsrelevanter Eisenbahnleitsysteme mit kooperierenden Werkzeugen, Dissertation, TU Braunschweig, 2002

[Bosch 2005] Fachartikel: Hybrid ist nicht gleich Hybrid, automotive – Fachmagazin für Entwicklungen in der Kfz-Elektronik und Telematik, WEKA Fachzeitschriften-Verlag GmbH, Poing, Heft 7 (November), 2005, S. 44 ff

[Computerbild 2003] Hitliste: Die schnellsten Prozessoren, Zeitschrift Computerbild, Axel Springer Verlag AG, Hamburg, Ausgabe 16/2003, S. 4

[Duden 1980] Der Große Duden, VEB Bibliographisches Institut Leipzig, 21. Auflage, 1980

[Elektronik 2003] Speicher-Spielchen: Wer ist der schnellste im ganzen Land, Fachzeitschrift Elektronik 16/2003, S. 26 ff

[EXITE 2004] EXITE Handbuch Version 1.3.0, EXTESSY AG, Wolfsburg, 2004

[EXITE 2005] EXITE Handbuch Version 1.4.0, EXTESSY AG, Wolfsburg, 2005,

[Elmqvist et al. 2002] H. Elmqvist, D. Brück, S. E. Mattson, H. Olson, M. Otter: Dymola, Dymola Modeling Laboratory – User's Manual, Dynasim AB, 2002

[Fritzson 2004] P. Fritzson: Principles of Object oriented Modeling and Simulation with Modelica 2.1, IEEE Press, 2004

[Fritzson et al. 2003] P. Fritzson (Editor): Proceedings of the 3rd International Modelica Conference, The Modelica Association, 2003

[Frontloading 2004] Erfolgsfaktor Frontloading, Fachartikel, Zeitschrift Konstruktionspraxis, Vogel Verlag, Würzburg, Ausgabe 04/2004

[Gaal 2003] Ruud van Gaal: Pacejka's Magic Formula
http://www.racer.nl/reference/pacejka.htm

[Gipser 2000] M. Gipser: Reifenmodelle in der Fahrzeugdynamik: eine einfache Formel genügt nicht mehr, auch wenn sie magisch ist, FH Esslingen, 2000

[GM 2005] GM – the low-level message-passing system for Myrinet networks,
http://www.myri.com/

[Götte et al.] Götte, T.; Pape, T.: DSG© – das Direktschaltgetriebe von Volkswagen, VDI-Berichte Nr. 1827: Getriebe in Fahrzeugen 2004, VDI Verlag GmbH, Düsseldorf, 2004

[Guest 2003] M. Guest: Application Performance on High-End and Commodity-class Computers, CCLRC Daresbury Laboratory, UK, 2003

[Hellestrand 2005] Graham Hellestrand: Schneller und preiswerter zum Ziel – Virtuelle Systemprototypen gewährleisten hohe Wiederverwendbarkeit von Entwicklungsplattformen und sparen Zeit und Geld, D&V Die Fachzeitschrift für Elektronik-Entwicklung, publish-industry Verlag GmbH, München, Ausgabe 05/2005, S. 23 ff

[Hommel 2005] Mathias Hommel: Der Golf ECO.Power – Ein zukunftsweisendes Hybridkonzept der Volkswagen Konzernforschung, Zeitschrift StandPUNKT, Verlag SG CONCEPTS, Wolfsburg, Ausgabe 01/2005

[Josefowitz et al. 2002] W. Josefowitz, S. Köhle: Aktivitäten bei Hybrid Antriebssystemen in Vergangenheit, Gegenwart und Zukunft und Beschreibung von Schlüsselkomponenten und deren Einfluß auf den Kraftstoffverbrauch, Symposium Hybridfahrzeuge / Energiemanagement, TU Braunschweig, 2002

[Kleinwechter et al. 2004] Konzernweite Modellierungsrichtlinien der Volkswagen AG für die Erstellung von Matlab/Simulink/Stateflow-Modellen, Version 1.2, Volkswagen AG, 2004

[Köhle 2004] S. Köhle: Der Volkswagen Bora Hybrid - Entwicklungsziele, Fahrzeugbeschreibung und erste Meßergebnisse des Volkswagen Bora mit Hybridantrieb, 2. Braunschweiger Symposium ‚Hybridfahrzeuge und Energiemanagement' , IHK Braunschweig, Braunschweig, 4.2.2004

[Kopetz et al. 1992] H.Kopetz, G.Grünsteidl: TTP – A Time Triggered Protocol for Automotive Applications. Forschungsbericht TU Wien Nr 16/1992

[Kopetz et al. 1993] H. Kopetz, G. Gruensteidl: TTP – A Time-Triggered Protocol for Fault-Tolerant Real-Time Systems, Proc. 23rd IEEE International Symposium on Fault-Tolerant Computing (FTCS-23), Toulouse, France, IEEE Press, 1993, (pp. 524-532), appeared also in a revised version in IEEE Computer. Vol. 24 (1). (pp. 22-26)

[Kovacevic 2002] S. Kovacevic: Distributed Software-in-the-loop Simulation of an Electronic Control Unit-Network Virtual Laboratory Vehicle, AEV GmbH, International Automotive Conference, Stuttgart, 2002

[Kramer 2002] U. Kramer: Modellbildung und Simulation des Systems Fahrer-Fahrzeug-Fahrumgebung, FH Bielefeld, 2001

[Kube et al.2004] Roland Kube, Michael Böckl, Mathias Hommel, Siegfried Köhle: Energy Management Strategies for Hybrid Drive Train Systems Using Infrastructure Information, EURmotor New Advances in Electronics Engineering, Energy Management – Today and tomorrow, 23.-24.9.2004, Aachen

[Leonhard 2000] W. Leonhard: Regelung elektrischer Antriebe, Springer Verlag, Heidelberg, 2. vollständig überarbeitete Auflage, 2000

[Lin et al. 2001] C.-C. Lin, Z. Filipi, Y. Wang, L. Louca, H. Peng, D. Assanis, J. Stein: Integrated, Feed-Forward Hybrid Electric Vehicle Simulation in Simulink and its Use for Power Management Studies, SAE 2001-01-1334, 2001

[Matlab 2002] MATLAB – The Language of Technical Computing, The Mathworks, Inc., Natick, USA, 2002

[Majumder et al. 2005] Supratik Majumder, Scott Rixner: Comparing Ethernet and Myrinet for MPI Communication, Rice University, Houston, Texas, USA, 2003

[Maxwell 2005] A. Schneuwly, J. Auer: Energiespeicher fürs Auto, automotive – Fachmagazin für Entwicklungen in der Kfz-Elektronik und Telematik, WEKA Fachzeitschriften-Verlag GmbH, Poing, Heft 7 (November), 2005, S. 72 ff

[Miegler et al. 2002] M. Miegler, W. Bauer-Kugelmann, G. Etzlstorfer, A. Schmitt: Das virtuelle Elektronik-Laborfahrzeug - Verteilte Echtzeitsimulation vernetzter Steuergeräte, Bertrand AG, 2002

[Miegler et al. 2003] M. Miegler, R. Gold, F. Großhauser: Simulationsbasierter Softwaretest von Steuergeräten – Ein Beispiel aus der Praxis, 22. Tagung „Elektronik im KFZ“, Stuttgart, 2002

[Mitschke 1990] M. Mitschke: Dynamik der Kraftfahrzeuge, Band C: Fahrverhalten, Springer Verlag, 1990

[Modelica 2004] The Modelica Association: The Modelica Language Specification http://www.modelica.org

[Modelica 2005] The Modelica Association: Modelica® - A Unified Object-Oriented Language for Physical Systems Modeling, Language Specification Version 2.2, February 2, 2005

[MPI 2005] MPI-Homepage: http://www-unix.mcs.nl.gov/mpi/

[MPICH 1998] Beschreibung und Download der MPI-1-Implementierung MPICH: http://www-unix.mcs.nl.gov/mpi/mpich/

[MPICH2 2005] Beschreibung und Download der MPI-2-Implementierung MPICH2: http://www-unix.mcs.nl.gov/mpich2/

[Mühlmeier 1993] M. Mühlmeier: Bewertung von Radlastschwankungen im Hinblick auf das Fahrverhalten von Pkw, Dissertation, Fortschrittberichte-VDI, Reihe 12, Nr. 187, VDI Verlag, Düsseldorf, 1993

[Myricom 2003] Myricom Inc.: Produktbeschreibung der Myrinet-Karte M3F2-PCIXE-2 http://www.myri.com/myrinet/PCIX/m3f-pcixd.html

[Otter et al. 2001] M. Otter, H. Elmqvist: Modelica – Language, Libraries, Tools, Workshop and EU-Project RealSim, Deutschen Zentrum für Luft- und Raumfahrt, Oberpfaffenhofen, 2001

[Otter et al. 2002] M. Otter (Editor): Proceedings of the 2nd International Modelica Conference, The Modelica Association, 2002

[Pacejka 2002] H. B. Pacejka, B. Heinemann: Tyre and Vehicle Dynamics, ASIN/ISBN: 0750651415

[Pape 2004] T. Pape: "Volkswagen DSG – Potential innovativer Automatikgetriebe mit Doppelkupplung", 16. Internationale AVL-Tagung Motor & Umwelt, 09. / 10. September 2004, Graz

[Pruckner et al. 2001] A. Pruckner, F. Noutsa Noufele, S. Fischer, M. Khajjou: PKW-Fahrdynamikreglerentwicklung mittels Zustandsraummodell am Beispiel eines 4WS-Prototypen, IKA Aachen, 2001

[Reimpell 1986] J. Reimpell: Fahrwerktechnik: Grundlagen, VOGEL Buchverlag Würzburg, 1. Aufl., 1986

[Reimpell et al. 1986] J. Reimpell, P. Sponagel: Fahrwerktechnik: Reifen und Räder, VOGEL Buchverlag Würzburg, 1986

[Rothacker 1996] M. Rothacker: TCP/IP-Grundlagen, Ruhr-Universität Bochum, 1996

[Rustemeier et al. 2005] C. Rustemeier, X. Liu-Henke, A. Goldau, K.-P. Jäker: Simulationsumgebung zur Modellkopplung von Black-Box-Modellen mechatronischer Funktionsmodule und MKS-Fahrzeugmodellen, VDI-Berichte Nr. 1892, 2005, VDI-Tagung „Mechatronik 2005“, Wiesloch, 1./2. Juni 2005

[Soejima 2000] S. Soejima: Examples of usage and spread of Dymola within Toyota, Modelica Workshop 2000, Lund, Sweden, Proceedings pp. 55-60

[Soejima et al. 2002] S. Soejima, T. Matsuba: Application of mixed mode integration and new implicit inline integration at Toyota, 2nd Int. Modelica Conference 2002, Proceedings pp. 65-1 - 65-6

[Supermicro 2004] Supermicro Computer, Inc., http://www.supermicro.com

[The Math Works Inc. 2004] The Math Works Inc: Using Simulink, http://www.mathworks.com

[Tiller et al. 2003a] M. Tiller, D. Linzen: A Comparison of Different Methods for Battery and Super capacitor Modelling, SAE 2003-01-2290, 2003

[Tiller et al. 2003b] M. Tiller, P. Bowles, M. Dempsey: Development of a Vehicle Modeling Architecture in Modelica, 3. Int. Modelica Conference, Linköping, Schweden, 2003

[TOP500] Lister der Top 500 Supercomputer: http://www.top500.org

[UniVW 2004] UniVW V6.0, SR1, Datenvisualisierung, Robert Bosch GmbH, Stuttgart, 2004

[V-Modell 1997] Die neue Version des V-Modells, http://www.vorgehensmodelle.de/Ws1997/tagungsband/vm-ext.doc

[V-Modell 2001] Software Testing Website – Das V-Modell; http://www.softwaretesting.de/article/view/36/1/7/

Anhang A

Performance of the PCIX-series PCI-bus-implementation

(nach [Myricom 2003])

All of the PCIX-series NICs use the same silicon implementation for the PCI/PCI-X logic, and exhibit the same performance in PCI DMA benchmarks.

The limit of a 64-bit, 133.3MHz PCI-X bus is 1067 MB/s, either reading or writing. In hosts with 64-bit, 100MHz, PCI-X buses, the limit is 800MB/s. The PCIX-series NICs achieve these data rates in bursts up to 4KB, the maximal DMA-transfer size for PCI-X, and perform all PCI-X bus protocols between bursts in a minimum of bus clock cycles.

The PCI-X slots in host computers transfer data to and from system memory, and thus can only approach the limit. For example, the host will typically delay the beginning of DMA-read transfers for ~60 PCI-X clock cycles while it starts fetching and buffering data from the host memory. The following table provides measurements of the PCI-DMA performance, as shown by the GM-2 "gm_debug" utility, of a sample of today's best cluster hosts. The test reports the average data rate for 32 chained 4KB reads to the same block, followed by 32 chained 4KB writes to the same block. **Note: Small differences are not significant.**

Host/OS	*bus read (send)*	*bus write (recv)*
AMD "Melody" dual 1.6GHz Opteron server (AMD 8131 chip set) / SuSE 8 Linux	936 MB/s	1032 MB/s
AMD "Quartet" quad Opteron server / SuSE 8 Linux	870 MB/s	989 MB/s
Apple dual 2GHz G5 / MacOS X	882 MB/s	1036 MB/s
HP "Marvel" (Alpha EV-7, es47) quad-Alpha server / either Linux or Tru64	908 MB/s	1038 MB/s
HP rx2000 dual 900MHz Itanium2 (HP chip set) / Linux	784 MB/s	1044 MB/s
IBM BladeCenter HS20 XEON blade, D-card HCA, 100MHz PCI-X / Linux	716 MB/s	784 MB/s
Intel quad 900MHz Itanium2 (Intel 870 chip set) / Linux	819 MB/s	947 MB/s
Intel dual 1.5GHz Itanium2 Madison / Linux	874 MB/s	946 MB/s
Intel dual 2.4GHz XEON whitebox (Serverworks GC chipset, 400MHz FSB) / Linux	856 MB/s	1044 MB/s
Intel dual 1.8GHz XEON whitebox (Intel E7500 chipset, 400MHz FSB) / Linux	816 MB/s	853 MB/s
Microway Navion dual 1.6GHz Opteron (AMD 8131 chip set) / United Linux kernel	939 MB/s	1036 MB/s
Newisys dual 1.4GHz Opteron server (AMD 8131 chip set) / SuSE 8 Linux	929 MB/s	1032 MB/s
Sunfire v60x dual XEON server (Intel E7500 chip set, 100MHz PCI-X) / Linux	675 MB/s	782 MB/s
Supermicro X5DL8-GG dual 2.4GHz XEON (Serverworks GC-LE chip set, 533MHz FSB) / Linux	932 MB/s	1044 MB/s
Supermicro X5DPE-G2 dual 2.4GHz XEON (Intel E7501 chip set, 533MHz FSB) / Linux	826 MB/s	853 MB/s
Tyan Trinity single 3.06GHz Pentium-4 (Serverworks GC-SL chip set, 533MHz FSB) / Linux *performance in the 133MHz PCI-X slot*	859 MB/s	1040 MB/s
Tyan Trinity single 3.06GHz Pentium-4 (Serverworks GC-SL chip set, 533MHz FSB) / Linux *performance in the 100MHz PCI-X slot*	708 MB/s	782 MB/s

How much bus performance do you need?

With the one-port NICs, 500 MB/s PCI-DMA performance is sufficient to achieve maximal summed-bidirection performance of 250+250 MB/s on the Myrinet port. For these one-port NICs, all of the hosts listed above have PCI-DMA performance to spare.

The two ports of the M3F2-PCIXE NICs have an aggregate peak throughput of 500MB/s from the Myrinet fabric plus 500MB/s to the fabric, a total of 1GB/s. The user-level summed-bidirectional data rate that GM-2.1 or MX-2G-Beta achieves is ~800MB/s, limited by NIC firmware and memory bandwidth. With balanced bidirectional traffic -- an equal number of bytes read and written, many of the hosts listed above have sufficient PCI-X bus performance to support ~800MB/s summed-bidirectional data rates. For example, the Supermicro X5DL8-GG hosts show 932MB/s bus read and 1044MB/s bus write. With an equal number of bytes read and written, the peak bus performance is 2/(1/932 + 1/1044) = 985 MB/s. (Note: To understand this formula, observe that 1/932 is the time to read a byte, and 1/1044 is the time to write a byte. In balanced traffic, two bytes are transferred each (1/932 + 1/1044) μs.)

Anhang B

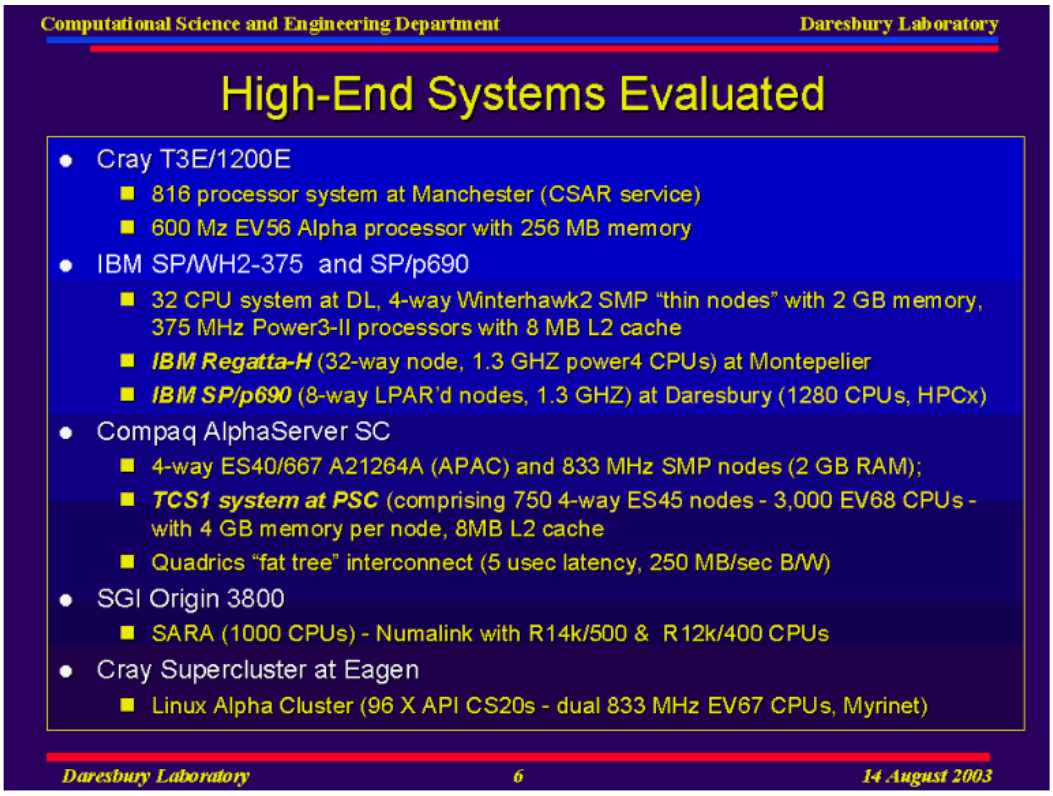
Ausdruck der Release Notes zum GM-Treiber, Version 2.1.x:

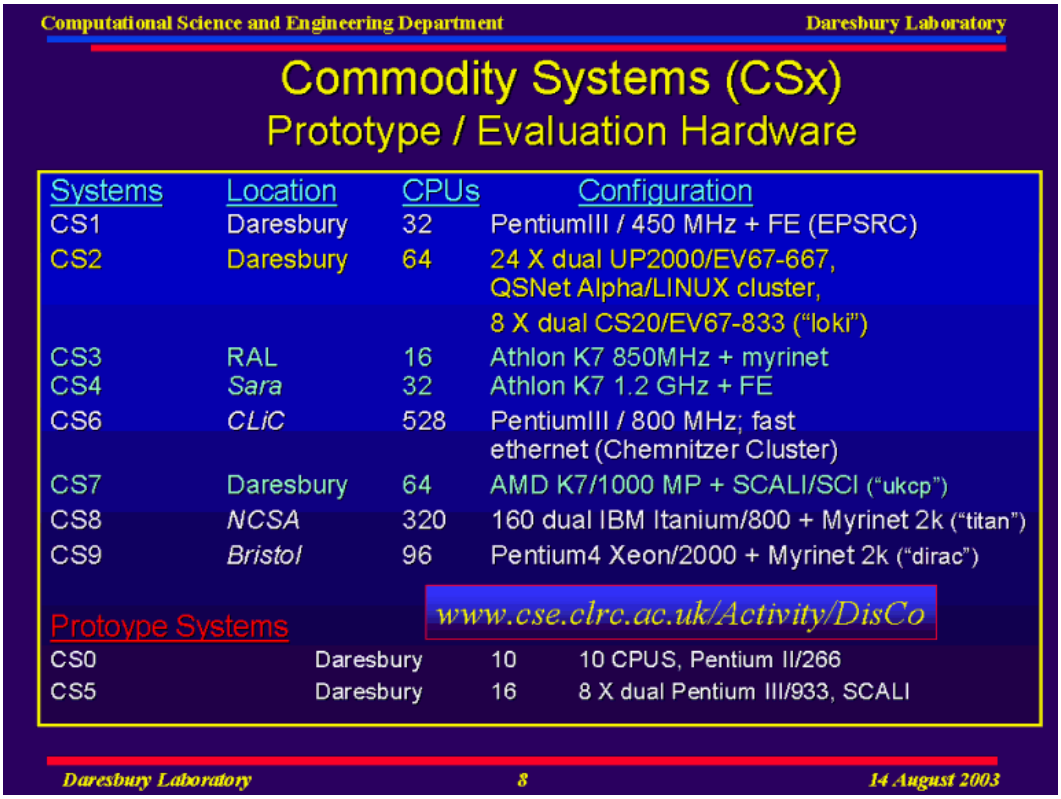
```
Release Notes for GM 2.1.0
==========================

This is release 2.1.0 of GM for Myricom PCIXD and PCIXE cards.  It is
the first release to support route dispersion and both ports of PCIXE
interfaces.

As usual, the performance you will observe depends on the machine in
which your Myrinet cards are installed.  In our lab on PCIXE cards, we
observe 5.42us minimum latency with
  host1> gm_allsize --size=20 --slave
  host2> gm_allsize --size=20 -h host1
We reproduce 796MB/s peak bidirectional bandwidth with
  host1> gm_allsize --size=20 --slave
  host2> gm_allsize --size=20 -h host1 -bw --both-ways --min-length=40960 \
        --length-increment=4096
and we reproduce 495MB/s unidirectional bandwidth with
  host1> gm_allsize --size=20 --slave
  host2> gm_allsize --size=20 -h host1 -u -bw -g

While this release supports redundant routes, it uses them in
round-robin fashion.  This provides some statistical relief from
network congestion hot spots, but we are developing alternative
strategies that promise even better load balancing.

Caveats:
========

This release has somewhat limited scalability, especially on PCIXE
cards because of the buffering requirements of the 2 packet
interfaces.  The scalability of this release is as follows:

        Myrinet Interfaces Supported on PCIX interfaces

                4K      16K     <- OS page size
              +-------------
        PCIXD | 1187    464
        PCIXE | 672

The scalability will be improved significantly in the next release.
Please contact help@myri.com if you have an urgent need for greater
scalability.

--

The output of gm_board_info has changed to represent multiple routes
over multiple packet interfaces.

--

Currently, Myrinet port N of each network interface card can
communicate only with Myrinet port N on remote NICs (except for
mapping packets).  Port 0 on one card cannot communicate with port 1
on a remote card.  This restriction may be relaxed in a future
release.  Take care if you are connecting the two ports of a PCIXE
interface to separate Myrinets.
```

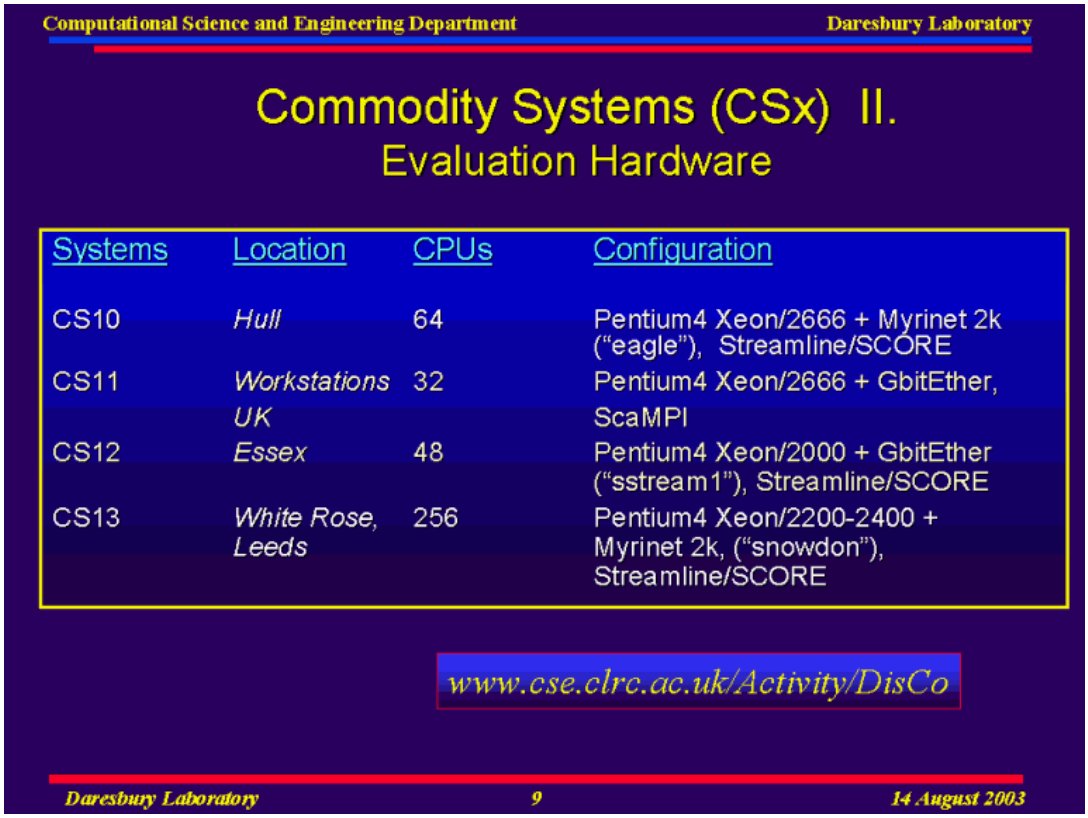
Anhang C

Rechnerkonfiguration beim Performancetest gemäß [Guest 2003]

(alle Bilder aus [Guest 2003])






Computational Science and Engineering Department
Daresbury Laboratory
Commodity Systems (CSx) II.
Evaluation Hardware
Systems
Location
CPUs
Configuration
CS10
Hull
64
Pentium4 Xeon/2666 + Myrinet 2k ("eagle"), Streamline/SCORE
CS11
Workstations UK
32
Pentium4 Xeon/2666 + GbitEther, ScaMPI
CS12
Essex
48
Pentium4 Xeon/2000 + GbitEther ("sstream1"), Streamline/SCORE
CS13
White Rose, Leeds
256
Pentium4 Xeon/2200-2400 + Myrinet 2k, ("snowdon"), Streamline/SCORE
www.cse.clrc.ac.uk/Activity/DisCo
Daresbury Laboratory
9
14 August 2003